НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

Кафедра системного програмування і спеціалізованих комп'ютерних систем

До захисту допущено

Завідувач кафедри
_____ Віталій РОМАНКЕВИЧ
(підпис) (ініціали, прізвище)

"___" _____ 2020 р.

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Системне програмування»

спеціальності **123 «Комп'ютерна інженерія»**

на тему: Засоби для знешумлення зображень з використанням нейронної мережі
_____
_____

Виконав :

студент IV курсу, групи КВ – 62
(шифр групи)

Ясенко Лев Сергійович _____ _____
(прізвище, ім'я, по батькові) (підпис)

Керівник доц.каф.СПСКС, к.т.н. Клятченко Я.М. _____ _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант з нормоконтролю, доц.каф.СПСКС, к.т.н. Клятченко Я.М. _____
(назва розділу) (посада, вчене звання, науковий ступінь, прізвище, ініціали) (підпис)

Рецензент_____ _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2020 року

ВІДОМІСТЬ ДИПЛОМНОГО ПРОЄКТУ

№ з/п	Формат	Позначення	Найменування	Кількість листів	Примітка
1	А4		Завдання на дипломний проєкт	2	
2	А4	ІАЛЦ 045480.001 ОА	Опис альбому	2	
3	А1	ІАЛЦ 045480.002 ТЗ	Технічне завдання	4	
4	А1	ІАЛЦ 045480.003 ТП	Відомість технічного проєкту	2	
5	А1	ІАЛЦ 045480.004 ПЗ	Пояснювальна записка	52	
6	А1	ІАЛЦ 045480.005 Д1	Структурна схема	1	
7	А1	ІАЛЦ 045480.006 Д2	Структурна схема	1	
8	А1	ІАЛЦ 045480.007 Д3	Схема алгоритму	1	
9	А1	ІАЛЦ 045480.008 Д4	Схема алгоритму	1	
10	А1	ІАЛЦ 045480.009 Д5	Схема алгоритму	1	
11	А1	ІАЛЦ 045480.010 Д6	Схема алгоритму	1	

				ІАЛЦ 045480.000		
	ПІБ	Підп.	Дата			
Розробн.	Ясенко Л.С			Відомість дипломного проєкту	Лист	Листів
Керівн.	Клятченко Я.М.				1	1
Консульт.					КПІ ім. Ігоря Сікорського Каф. СПіСКС Гр. КВ-62	
Н/контр.	Клятченко Я.М.					
Зав.каф.	Романкевич В.О.					

Пояснювальна записка
до дипломного проєкту

на тему: Засоби для знешумлення зображень з використанням нейронної мережі
_____
_____

Київ – 2020 року

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

Кафедра системного програмування і спеціалізованих комп'ютерних систем

Рівень вищої освіти – перший (бакалаврський)

Спеціальність 123 «Комп'ютерна інженерія»

Освітньо-професійна програма «Системне програмування»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Віталій РОМАНКЕВИЧ
(підпис) (ініціали, прізвище)
«___» _____20___ р.

ЗАВДАННЯ
на дипломний проєкт студента

Ясенко Лев Сергійович
(прізвище, ім'я, по батькові)

1. Тема проєкту «Засоби для знешумлення зображень з використанням нейронної мережі» _____
_____ ,
керівник проєкту Клятченко Ярослав Михайлович доц.каф.СПСКС, к.т.н. ____ ,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «__» _____ 2020 р. №_____

2. Термін подання студентом проєкту _____

3. Вихідні дані до проєкту: розроблена програма засобу для знешумлення зображень з використанням нейронної мережі, набори зображень._____

4. Зміст пояснювальної записки: перелік термінів та скорочень; вступ; розділ 1. Узагальнена модель зображень сформованих під час рендерингу; розділ 2. Нейронна мережа в якості фільтра цифрових зображень; розділ 3. Опис програмного засобу; розділ 4. Перевірка рішення; Висновки; Перелік використаних джерел. _____

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслеників, плакатів, презентацій тощо): нейронна мережа для знешумлення зображень. Структурна схема; взаємозв'язки модулів програмного засобу. Структурна схема; сегментація зображення. Схема алгоритму; відновлення зображення з сегментів. Схема алгоритму; тренування нейронної мережі для знешумлення. Схема алгоритму; тестування нейронної мережі для знешумлення. Схема алгоритму; тренувальні і тестові зображення; 28 рисунків; презентація. _____

6. Консультанти розділів проєкту*

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання _____

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1	Видача завдання на дипломне проєктування	19.12.2019	
2	Вивчення літератури	26.12.2019	
3	Розробка та узгодження технічного завдання	22.01.2020	
4	Створення зображень для обробки	03.02.2020	
5	Розробка структури програмного засобу	09.02.2020	
6	Розробка дизайну програмного засобу	30.03.2020	
7	Тестування програмного засобу	28.04.2020	
8	Підготовка матеріалів дипломного проєкту	18.05.2020	
9	Оформлення технічної документації	02.06.2020	

Студент _____ Лев ЯСЕНКО
(підпис) (Ім'я та ПРІЗВИЩЕ)

Керівник проєкту _____ Ярослав КЛЯТЧЕНКО
(підпис) (Ім'я та ПРІЗВИЩЕ)

* Консультантом не може бути зазначено керівника дипломного проєкту.

Анотація

Об’єкт розробки — це програмний засіб цифрової обробки зображень, який дозволяє за допомогою нейронної мережі створювати адаптивні фільтри для них.

Предмет розробки — фільтрація випадкових шумів (артефактів).

Цей засіб дозволяє користувачам завантажувати власні зображення, тренувати, тестувати та зберігати ваги нейронної мережі, яка виконує роль адаптивного фільтра шумів. В процесі розробки використано скриптову мову програмування Python та бібліотеки OpenCV, NumPy, PyTorch, MatPlotLib та pyforms.

В ході розробки:

– описана модель шумів (артефактів), які виникають в ході формування зображення шляхом рендерингу в програмних засобах моделювання тривимірних сцен;

– показана загальна подібність таких викривлень з особливостями формування цифрових зображень реальних систем спостереження (астрономічні, рентгенівські, тощо);

– проведено аналіз відомих типів нейронних мереж на придатність до застосування в задачах знеструмлення цифрових зображень;

– вибрано середовище розробки;

– розроблено програмний засіб обробки (знешумлення) цифрових зображень з користувацьким інтерфейсом на основі нейронної мережі;

– проведено дослідження ефективності розробленого програмного засобу.

Ключові слова: рендеринг, знешумлення, нейронна мережа, цифрове зображення, тривимірна сцена, шум.

Abstract

The object of development is a software tool for digital image processing, which allows using a neural network to create adaptive filters for images.

The subject of development is the filtering (denoising) of random noise (artifacts).

This tool allows users to upload their own images, train, test and store the weights of the neural network, which acts as an adaptive noise filter. In the development process were using scripting programming language Python and such libraries as OpenCV, NumPy, PyTorch, MatPlotLib and pyforms.

During development:

- described a model of noise (artifacts) that occur during image (based on a 3D scene) rendering process;
- the general similarity of such distortions with digital images formation features of real supervision systems (astronomical, X-ray, etc.) is shown;
- the analysis of known neural networks types on suitability for solving the problem of digital images denoising is carried out;
- the development environment is selected;
- developed a software tool with a user interface for digital image processing (noise reduction) based on a neural network;
- studied the efficiency of the developed software.

Keywords: rendering , denoising, neural network, digital image, 3D scene, noise.

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
	А4	ІАЛЦ.045480.002 ТЗ	Засоби для знешумлення	4		
			зображень з використанням			
			нейронної мережі			
			Технічне завдання			
	А4	ІАЛЦ.045480.003 ТП	Засоби для знешумлення	2		
			зображень з використанням			
			нейронної мережі			
			Відомість технічного			
			проєкту			
	А4	ІАЛЦ.045480.004 ПЗ	Засоби для знешумлення	52		
			зображень з використанням			
			нейронної мережі			
			Пояснювальна записка			
	А1	ІАЛЦ.045480.005 Д1	Нейронна мережа для	1		
			знешумлення зображень.			
			Структурна схема			
	А1	ІАЛЦ.045480.006 Д2	Вмодулів програмного	1		
			засобу.			
			Структурна схема			

					ІАЛЦ.045480.001 ОА		
Змін.	Арк.	№ докум.	Підпис	Дата			
Розробив		Ясенко Л.С.			Засоби для знешумлення зображень з використанням нейронної мережі. Опис альбому	Літ.	Арк 1 / Арк 2
Перевірив		Клятченко Я.М.					
Консульт.						КПІ ім. Ігоря Сікорського, ФПМ гр. КВ-62	
Н. контроль		Клятченко Я.М.					
Зав. каф.		Романкевич В.О.					

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
	А1	ІАЛЦ.045480.007 Д3	Сегментація зображення.	1		
			Схема алгоритму			
	А1	ІАЛЦ.045480.008 Д4	Відновлення зображення з	1		
			сегментів.			
			Схема алгоритму			
	А1	ІАЛЦ.045480.009 Д5	Тренування нейронної	1		
			мережі для знешумлення.			
			Схема алгоритму			
	А1	ІАЛЦ.045480.010 Д6	Тестування нейронної	1		
			мережі для знешумлення.			
			Схема алгоритму			
		Диск CD-ROM	Матеріали дипломного	1		
			проєкту.			

Змін.	Арк.	№ докум.	Підпис	Дата	ІАЛЦ.045480.001 ОА	Арк. 2

ЗМІСТ

1 НАЙМЕНУВАННЯ ТА ГАЛУЗЬ РОЗРОБКИ

Назва розробки: “Засоби для знешумлення зображень з використанням нейронної мережі”.

Галузь застосування: створення системи обробки цифрових зображень з допомогою нейронної мережі.

2 ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки є завдання на виконання роботи першого (бакалаврського) рівня вищої освіти, затверджене кафедрою системного програмування і спеціалізованих комп’ютерних систем Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського».

3 МЕТА І ПРИЗНАЧЕННЯ РОЗРОБКИ

Метою даного проєкту є створення програмного засобу для управління системою обробки зображень, шляхом задання параметрів користувачем.

4 ДЖЕРЕЛА РОЗРОБКИ

Джерелом інформації є технічна та науково-технічна література, технічна документація, публікації в періодичних виданнях та електронні статті у мережі Інтернет.

5 ТЕХНІЧНІ ВИМОГИ

5.1 Вимоги до програмного продукту, що розробляється

- Можливість завантаження довільних зображень визначеного формату (png).
- Можливість зберігати та завантажувати ваги нейронної мережі.
- Можливість проводити тестування на окремих зображеннях.
- Забезпечення виводу інформації для користувача у вікні програмного засобу для користувача.
- Забезпечити службовий вивід інформації в терміналі.

5.2 Вимоги до апаратного забезпечення

- Наявність центрального або графічного процесора.
- Наявність оперативної та постійної пам’яті.

5.3 Вимоги до програмного забезпечення користувача

- Наявність інтерпритатора python та бібліотек numpy, opencv, pytorch та pyforms.

6 ЕТАПИ РОЗРОБКИ

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів
1.	Видача завдання на дипломне проєктування	19.12.2019
2.	Вивчення літератури	26.12.2019
3.	Розробка та узгодження технічного завдання	22.01.2020

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів
4.	Створення зображень для обробки	03.02.2020
5.	Розробка структури програмного засобу	09.02.2020
6.	Розробка дизайну програмного засобу	30.03.2020
7.	Тестування програмного засобу	28.04.2020
8.	Підготовка матеріалів дипломного проєкту	18.05.2020
9.	Оформлення технічної документації	02.06.2020

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
	А4	ІАЛЦ.045480.004 ПЗ	Засоби для знешумлення	52		
			зображень з використанням			
			нейронної мережі			
			Пояснювальна записка			
	А1	ІАЛЦ.045480.005 Д1	Нейронна мережа для	1		
			знешумлення зображень.			
			Структурна схема			
	А1	ІАЛЦ.045480.006 Д2	Вмодулів програмного	1		
			засобу.			
			Структурна схема			
	А1	ІАЛЦ.045480.007 Д3	Сегментація зображення.	1		
			Схема алгоритму			
	А1	ІАЛЦ.045480.008 Д4	Відновлення зображення з	1		
			сегментів.			
			Схема алгоритму			
	А1	ІАЛЦ.045480.009 Д5	Тренування нейронної	1		
			мережі для знешумлення.			
			Схема алгоритму			

Змін	Арк.	№ докум.	Підпис	Дата	ІАЛЦ.045480.003 ТП		
Розробив		Ясенко Л.С.			Засоби для знешумлення зображень з використанням нейронної мережі. Відомість технічного проєкту	Літ.	Аркуш 1 / Аркуші 2
Перевірив		Клятченко Я.М.					
Консульт.						КПІ ім. Ігоря Сікорського, ФПМ КВ-62	
Н. контроль		Клятченко Я.М.					
Зав. каф.		Романкевич В.О.					

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
	А1	ІАЛЦ.045480.010 Д6	Тестування нейронної	1		
			мережі для знешумлення.			
			Схема алгоритму			
		Диск CD-ROM	Матеріали дипломного	1		
			проєкту.			

Змін.	Арк.	№ докум.	Підпис	Дата	ІАЛЦ.045480.003 ТП	Арк. 2

ЗМІСТ

ПЕРЕЛІК ТЕРМІНІВ ТА СКОРОЧЕНЬ

ЗНМ – згорткова нейронна мережа;

НМ – нейронна мережа;

ПЗЗ – прилади з зарядовим зв’язком;

рендеринг – процес візуалізації графічних об’єктів за моделлю;

фреймворк – програмне забезпечення, що полегшує процес розробки програм, забезпечує можливість вибіркових змін користувацьким текстом програм.

ядро згортки – функція, що описує характеристику просторового фільтру;

DAE – denoising autoencoder (спеціальна нейронна мережа для знежумлення даних);

RGB – спосіб кодування зображень (R - червоний, G - зелений, B - синій);

ВСТУП

Однією з проблем створення реалістичних зображень в комп'ютерній графіці за допомогою спеціальних програмних засобів візуалізації 3D моделей є використання великої кількості ресурсів та часу. Недостатній час обробки, проявляється на зображення структурою, подібною до шуму. Щоб скоротити час візуалізації можна збільшити кількість ресурсів. Типовим рішенням є використання кількох центральних процесорів та (або) графічних карт. Разом з тим, таке рішення вимагає значного збільшення фінансових витрат. Тому продовжується пошук інших способів покращення зображень за рахунок програмної обробки. Останнім часом в джерелах часто згадують обробку зображень з використанням нейронних мереж (НМ).

Обробка зображень за допомогою НМ подібна до обробки за допомогою класичних фільтрів. Параметри НМ формуються в режимі тренування, з подальшим застосуванням її в якості фільтра. Такий підхід дозволяє отримати фільтр адаптований до конкретних зображень. Таким чином, задача побудови структури і підбору параметрів НМ, здатної адаптивно навчатися формуванню фільтрів для обробки зображень, є актуальною.

РОЗДІЛ 1. УЗАГАЛЬНЕНА МОДЕЛЬ ЗОБРАЖЕНЬ СФОРМОВАНИХ В ПРОЦЕСІ РЕНДЕРИНГУ

1.1 Узагальнена модель цифрових зображень

Як правило, цифрові зображення у двовимірному просторі представляються за допомогою двовимірної функції від координат F(x, y) [1]. Особливості цифрових зображень, викликані природою технічних систем зору з використанням обчислювальних пристроїв та матричних приймачів зображень. Більш подібними до проєкцій розподілу квантів випромінювання в реальному світі є аналогові зображення. У випадку, якщо в системах тривимірного моделювання, модель сцени описано за допомогою математичних формул, векторів, така модель близька до реальних об’єктів, а кінцеві зображення є цифровими. З урахуванням мети даної роботи, розглянемо особливості опису цифрових зображень. Припустимо, що зображення нескінчених розмірів описується формулою F(x, y). В ідеальній системі дискретизації просторові відліки характерного параметра зображення утворюють шляхом множення функції, що описує цей параметр на функцію дискретизації. Для здійснення дискретизації застосовують наступну формулу:

$$\mathrm{D}(x,y) = \sum_{m=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} \delta(x - m\Delta x, y - n\Delta y), \tag{1}$$

яка утворює решітку дискретизації, що містить δ-функції озташовані з кроком Δx, Δy у відповідних напрямках.

Дискретизоване зображення можна описати таким співвідношенням:

$$F_p(x,y) = F(x,y)D(x,y) = \sum_{m=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} F(m\Delta x, n\Delta y)\delta(x - m\Delta x, y - n\Delta y), \tag{2}$$

де x та y — координати, $F_p(x, y)$ — дискретизоване зображення [1].

Зображення можна розглядати як тривимірну випадкову функція деякої фізичної величини L(x,y,t). У кожний конкретний момент часу зображення можна розглянути як миттєвий знімок якоїсь сцени. У цьому випадку зображення можна розглядати як випадкове двовимірне поле й оцінити можливість подання такого поля дискретними відліками. Статистичний опис зображень широко використовують під час кодування джерела зображення й каналу передачі. Оскільки на підставі зробленого раніше зауваження випливає, що таке подання можливо зробити, тому зосередимо нашу увагу головним чином на характеристиках й функціональних залежностях, які використовують у цьому випадку[1].

З початку появи цифрових зображень, виникло питання забезпечення необхідної їх якості, це стосувалося і компенсації артефактів (шумів) імпульсної природи. Відтак, давно відомі класичні методи фільтрації таких завад. Основою класичних методів боротьби з завадами є просторова і частотна фільтрація [2].

1.2 Особливості тестових зображень

Модель зображень, що формуються в процесі рендерингу, визначається дискретною структурою сигналів, які обробляються в сучасних інформаційних системах і являє собою функцію розподілу яскравості спектральної складової сигналу зображення в просторі і часі, що можна представити формулою:

$$F_R(X, Y, Z, T) = \sum_{i=1}^{N} L_i(p_i, w_i), \qquad (3)$$

де L_0 — світіння джерела світла, що потрапляє в точку з координатами [X, Y, Z, T] з напрямку ω_i, і — номер джерела світла, p_i — точка на поверхні сцени з координатами [X, Y, Z], ω_i - напрямок вихідного світла, N — загальна кількість джерел світла.

В більшості випадків, сучасні змодельовані в ході рендерингу двовимірні (“пласкі”) зображення, що відповідають конкретному значенню

часу, які є аналогом двовимірних проєкцій три-вимірної сцени. Загалом формування зображення за моделлю описано формулою рендерингу:

$$L_0(p,\omega_0) = L_e(p,\omega_0) + \int_{2\pi} f_r(p,\omega_i,\omega_0)\, L_i(p,\omega_1) cos\theta_i d\omega_1, \qquad (4)$$

де p — точка на поверхні об'єктів сцени, ω_0 — напрямок вихідного світла, ω_i — напрямок вхідного світла, $L_0(p, \omega_0)$ — яскравість, що потрапляє в точці p, $L_0(p, \omega_0)$ — яскравість, що випромінюється в точці p, $f_r(p, \omega_i, \omega_0)$ — двопроменева функція здатності відбивання, $L_i(p, \omega_i)$ — випромінювання, що потрапляє в точку p, $cos\theta_i$ — задається точковим добутком між ω_i і нормаллю в точці p, і є коефіцієнтом ослаблення випромінювання через кут падіння [3].

Однак сучасні обчислювальні системи дозволяють, в межах конкретного часового інтервалу, обробити обмежену кількість променів (трас) для обмеженої кількості точок сцени. Це визначає основний тип спотворень (недоформованості) зображень, що заважають оператору (людині) сприймати такі зображення. Ці спотворення подібні за своїми характеристиками до імпульсних шумів. В реальному світі існують класи зображень, які мають схожі спотворення. Такі зображення, як правило отримуються за умови малої кількості джерел випромінювання (коли має значення кожен квант випромінювання) та (,або) невисокої роздільної здатності приймачів випромінювання. Прикладами систем, в яких отримані зазначені класи зображень є: астрономічні системи спостережень, системи технічної та медичної рентгенографії, інфрачервоні (тепловізійні) системи спостереження у певних умовах.

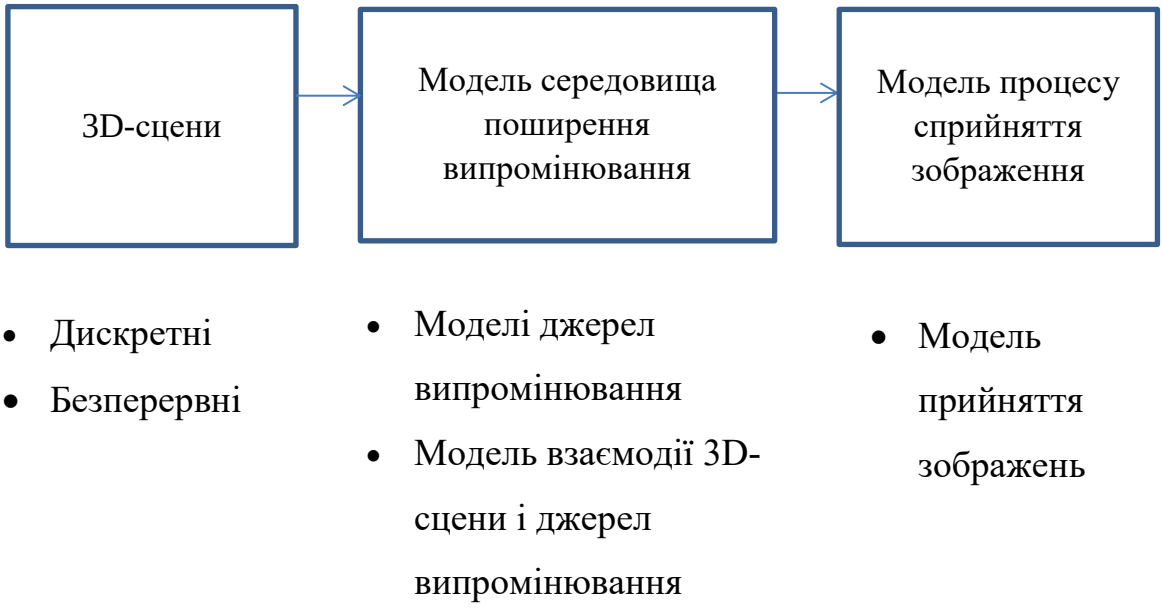


Рисунок 1 - Узагальнена модель отримання тестових зображень при рендерингу.

Терміном шум зазвичай називають такі дані, з яких неможливо (або невідомо як) добути інформацію, все інше називається сигналом. Не правильно вважати, що шум не містить жодної інформації. Шум можна змоделювати, для його опису. Суть шуму в тому, що ним не хочуть користуватися [4].

Основними джерелами шуму на цифрових зображеннях є сам процес його отримання, а також процес передачі. Для подальшого розгляду шуму, важливими є параметри, що визначають просторові характеристики шуму. Існують певні шуми для яких визначені функції густини розподілу ймовірностей [2]:

- Гауссів шум:

$$p(z) = \frac{1}{\sqrt{2\pi\sigma}} e^{-(z-\mu)^2/2\sigma^2}, \quad (5)$$

де z — це яскравість точки, μ — середнє значення випадкової величини z, σ — її середньоквадратичне відхилення,

- шум Релея:

$$p(z) = \begin{cases} \frac{2}{b}(z-a)e^{-(z-a)^2/b} & \text{при } z \geq a; \\ 0 & \text{при } z < a. \end{cases} \quad (6)$$

де a і b — це яскравості.

- шум Ерланга (гама шум):

$$p(z) = \begin{cases} \frac{a^b z^{b-1}}{(b-1)!} e^{-az} & \text{при } z \geq 0; \\ 0 & \text{при } z < 0. \end{cases} \quad (7)$$

- експоненційний шум:

$$p(z) = \begin{cases} ae^{-az} & \text{при } z \geq 0; \\ 0 & \text{при } z < 0. \end{cases} \quad (8)$$

- рівномірний шум:

$$p(z) = \begin{cases} \frac{1}{b-a} & \text{при } a \leq z \leq b; \\ 0 & \text{в інших випадках.} \end{cases} \quad (9)$$

- імпульсний шум:

$$p(z) = \begin{cases} P_a & \text{при } z = a; \\ P_b & \text{при } z = b; \\ 0 & \text{в інших випадках.} \end{cases} \quad (10)$$

Одним із характерних ознак шумів на зображеннях є їх гістограма. Гістограми описаних раніше шумів надано в [2]. Для з'ясування характеристики артефактів (шумів), які формуються в процесі рендерингу, було створено тестову тривимірну сцену, що складалася з плоских фігур, квадратної та круглої форми і джерела світла, зображення якої подібне до рисунку 5.3 у [2] і представлено рисунком 2.

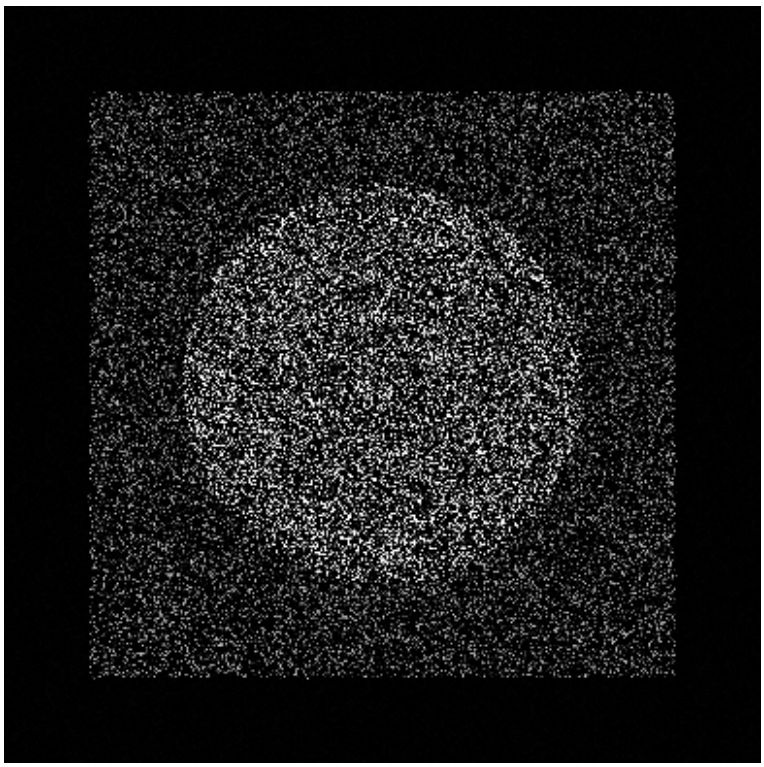


Рисунок 2 - Зображення проєкції тестової сцени, з чотирма вибірками.

Гістограми розподілу яскравості на отриманому зображені після рендерингу представлена на рисунку 3.

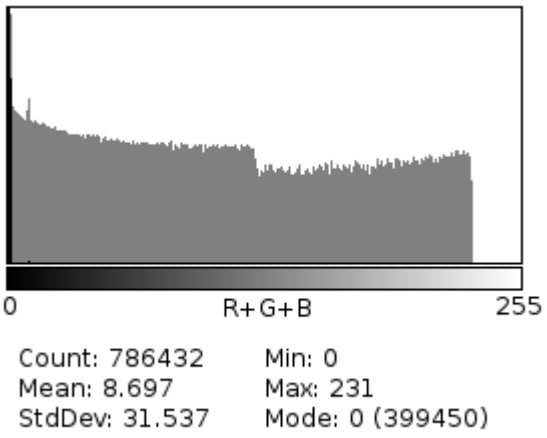
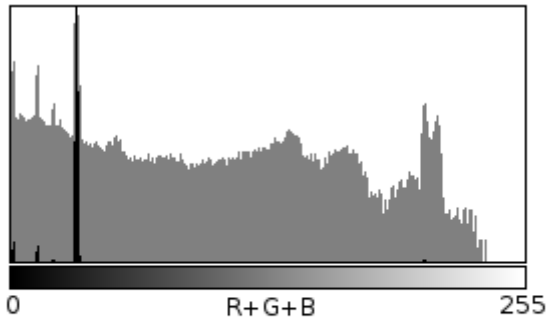


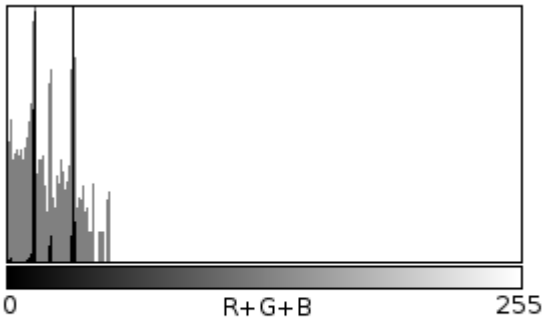
Рисунок 3 - Гістограми до рисунку 1 (чорний — лінійний масштаб, сірий — логарифмічний масштаб).

Для оцінки характеристик тестових зображень з шумом було виконано рендеринг тестової сцени (1 вибірка), та побудовано відповідні гістограми розподілу яскравості, що представлені на рисунках 4 - 5 .



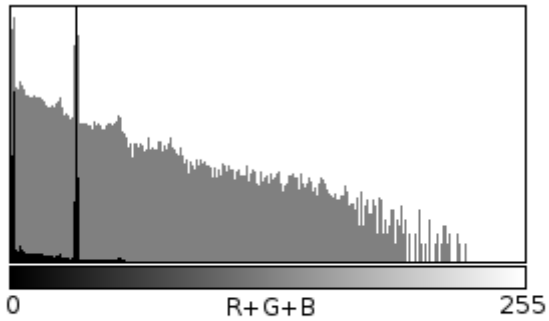
Count: 786432 Min: 0
Mean: 34.624 Max: 240
StdDev: 23.841 Mode: 32 (548862)

а



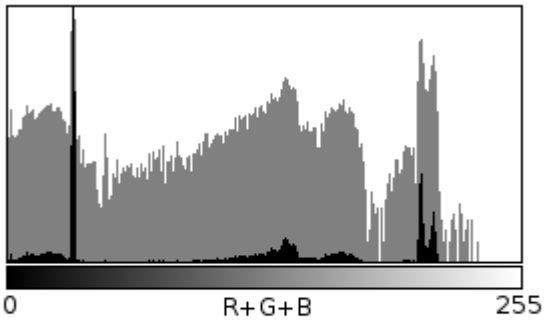
Count: 27324 Min: 0
Mean: 20.750 Max: 74
StdDev: 9.698 Mode: 32 (8547)

б



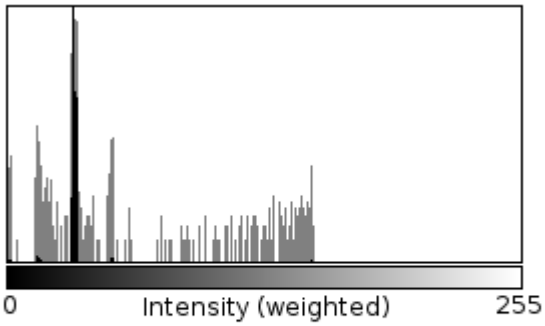
Count: 68808 Min: 0
Mean: 27.361 Max: 233
StdDev: 25.488 Mode: 32 (26717)

в



Count: 46269 Min: 0
Mean: 86.029 Max: 240
StdDev: 70.833 Mode: 32 (17310)

г



Count: 7050 Min: 0
Mean: 34.995 Max: 152
StdDev: 16.821 Mode: 32 (3767)

Д

Рисунок 4 - Гістограми розподілу тестового зображення: а — до всього рисунку, б — до фрагменту з кубом, в — до “мавпочки”, г — до площини, д — до сфери (чорний — лінійний масштаб, сірий - логарифмічний).

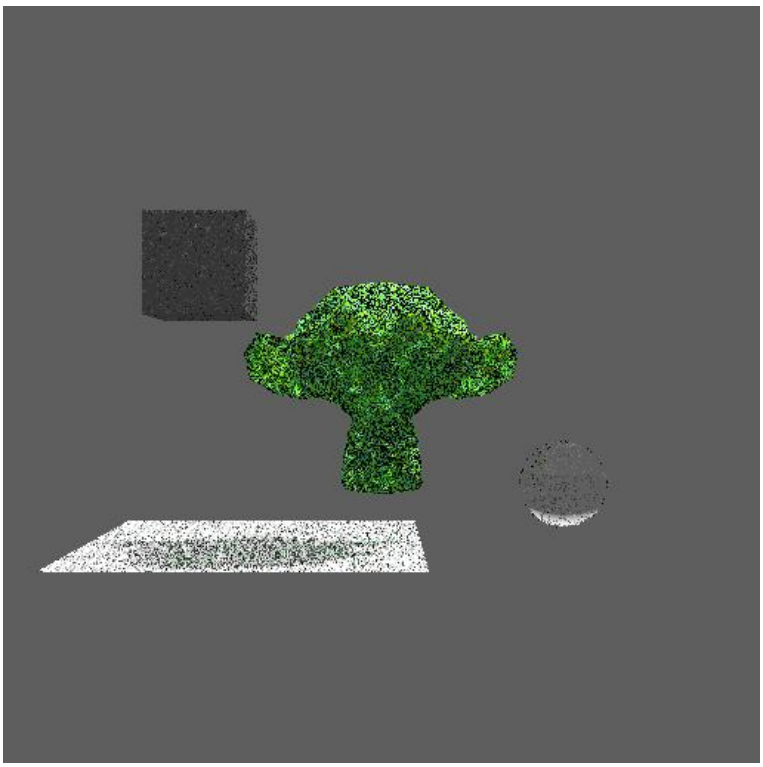


Рисунок 5 - Зображення тестової сцени (одна вибірка).

Аналізуючи отримані гістограми, можна сказати, що параметри суміші сигнал-шум для різних сегментів зображення має різну форму. Можна передбачити, що характеристики оптимального (квазі-оптимального) фільтра, для компенсації артефактів (шумів), мають бути адаптивні, при чому мінятися, як за площею зображення, так і від одного зображення до іншого. Підбір параметрів фільтрів в таких умовах є рутинною операцією, що вимагає значних ресурсів часу, зазвичай такі операції підлягають автоматизації. Одним з напрямків автоматизації є застосування НМ [2].

1.3 Висновки за розділ

Основою моделі зображень є модель цифрового (дискретизованого) зображення. В свою чергу, дане цифрове зображення є двовимірною проєкцією тривимірної сцени, яка згенерована алгоритмічним способом в процесі рендерингу.

Зображення отримані в процесі рендерингу, за умови не достатньої кількості часу обробки, характеризуються шумоподібною структурою, що

викликано особливостями процесу рендерингу. Статистичні характеристики такого шуму за площею зображення, можуть відповідати різним розподілам (Гаусса, Релея, Ерланга, експотенційного, рівномірного, імпульсного), що визначає необхідність застосування до таких зображень процедури адаптивної фільтрації.

Актуальною є побудова адаптивного фільтру на основі НМ, для знешумлення цифрових зображень отриманих в ході рендерингу, при не достатній кількості часу на рендеринг. Такий фільтр має компенсувати артефакти (шуми), на тестових зображеннях під час їх підготовки до сприйняття оператором.

РОЗДІЛ 2. НЕЙРОННА МЕРЕЖА В ЯКОСТІ ФІЛЬТРУ ЦИФРОВИХ ЗОБРАЖЕНЬ

2.1 Типи нейронних мереж

Питання створення обчислювальних середовищ, здатних обробляти великі об’єми даних, постало давно, включно з середовищами подібними до НМ. Прикладом простого середовища обробки візуальної інформації є шар простору зі збиральною лінзою, який виконує двовимірне перетворення Фур’є по всій площині перетворюваного зображення (див. с. 20 у [5]).

Другим поштовхом для розвитку технічних засобів, що можуть реалізовувати обчислювальні середовища, стала поява матричних приймачів зображень на основі приладів зі зарядовим зв’язком (ПЗЗ, див у [6], [7]). Причому, структури ПЗЗ, в одних мікросхемах забезпечували сприйняття візуальної інформації (світлочутливі елементи), а в інших – зберігання та обробку. З часом світлочутливі елементи стали поєднувати з елементами пам’яті та обробки. Ідея такого об’єднання була розширена до ідеї формування багатошарових систем сприйняття і обробки інформації на одному кристалів. Одним з основних алгоритмів обробки зображень в таких структурах є згортка. Такі багатошарові структури можна вважати апаратною реалізацією обчислювальних середовищ, подібних до НМ (див. с 173-175 в [8]).

Особливістю обчислювальних середовищ є реалізація однакового оператору обробки в площині одного шару (паралельна обробка), що дозволяє повторити таку операцію послідовно, застосовуючи обчислювальний пристрій, який виконує по одній операції за раз (послідовна обробка). Крім того, існують проміжні варіанти (послідовно-паралельна обробка). На вибір схеми обчислень впливає кількість даних до обробки і наявні ресурси системи. Паралельна, послідовна і паралельно-

послідовна схеми є прикладами віртуалізації обчислювальних середовищ. Таке віртуальне обчислювальне середовище спроможне обробити цифрові зображення, не залежно від способу їх отримання (отримання зображень реальних об’єктів, синтез штучних зображень).

Сьогодні існує певний перелік моделей НМ, розроблених для обробки зображень з використанням наявного апаратного забезпечення. Для формування програмної реалізації обчислювальних середовищ обробки зображень на основі НМ, необхідно визначити тип НМ та її параметри.

Для ефективної обробки зображень застосовують згорткові нейронні мережі (ЗНМ). Такі мережі базуються на математичній операції згортки. Загальний порядок фільтрації зображень за допомогою нейронної мережі складається з двох етапів, а саме: кодування та відновлення. В окремих випадках може бути застосований один з них. На етапі кодування виконується операції згортки та проріджування, що можуть бути об’єднані в одну або виконані послідовно. На етапі відновлення виконується інтерполяційна процедура, що своїм змістом протилежна проріджуванню, а за формою схожа на операцію згортки (функція ядра згортки відповідає інтерполяційній функції). Етапи складаються з повторюваних за кількістю шарів НМ операцій кодування та відновлення.

Операція згортки застосовується до вхідного шару НМ, а результат передається до наступного шару. Така організація НМ імітує реакцію біологічних нейронів на візуальні подразники [9].

$$C_{out_j} = B_j + \sum_{k=0}^{N-1} W_{Out_{j,k}} * C_{in_k}, \qquad (11)$$

де C_{out} – канал шару згортки на виході, B – початкове значення, N – кількість каналів вхідного шару, W – вага (фільтр), C_{in} – канал шару згортки на вході [10].

Алгоритм формування значень каналів в новому шарі шляхом виконання операції згортки проілюстровано на рисунку 6.

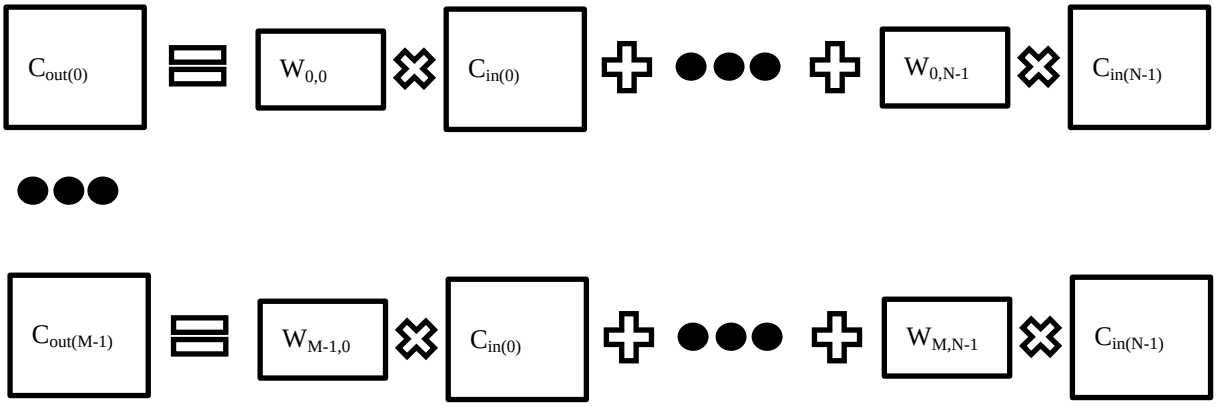
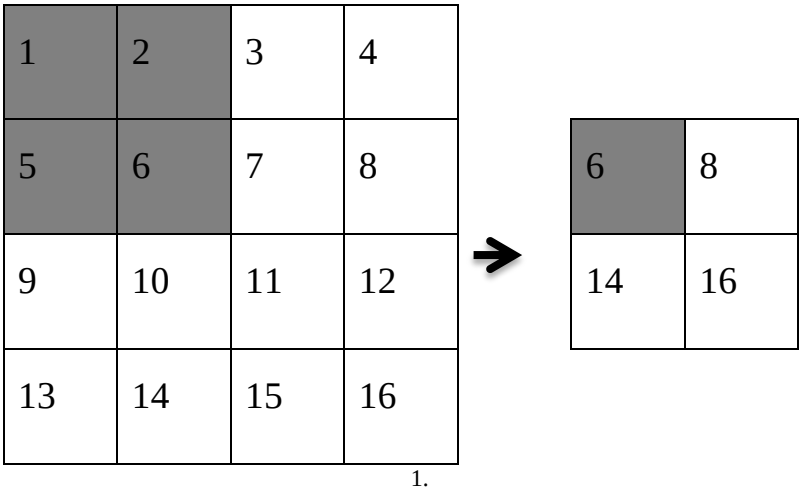


Рисунок 6 - Отримання нових каналів виходу шару згортки: M – кількість вихідних каналів, N – кількість вхідних каналів, W – фільтр згортки, C – канал.

В результаті виконання операції прорідження кількість каналів в шарі залишається незмінною, а кількість елементів розкладення зменшується з певною кратністю (с 107-108 у [2]). Приклад такої операції з використанням фільтру максимуму зображено на рисунку 7 та 8.



1.

Рисунок 7 - Приклад функції MaxPool2D з пакету PyTorch (фільтр максимуму).

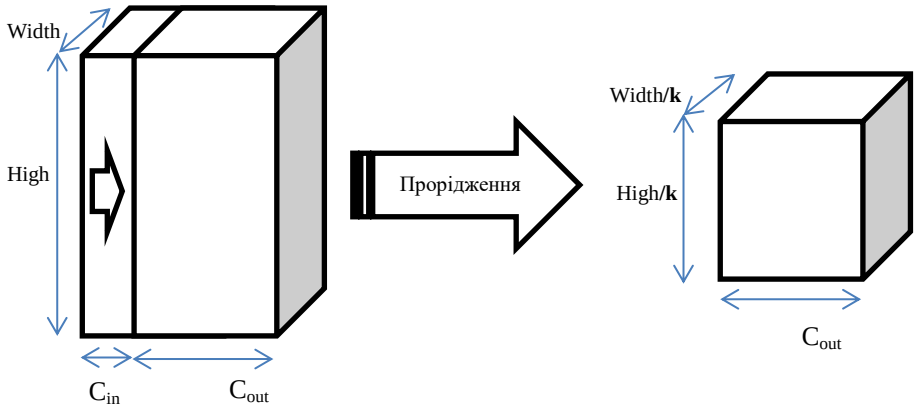


Рисунок 8 - Результат прорідження: High*Width формують площу каналу в шарі, C_{in} – вхідні канали, C_{out} – вихідні канали, k – коефіцієнт прорідження.

2.

Операція відновлення проілюстрована на рисунку 9.

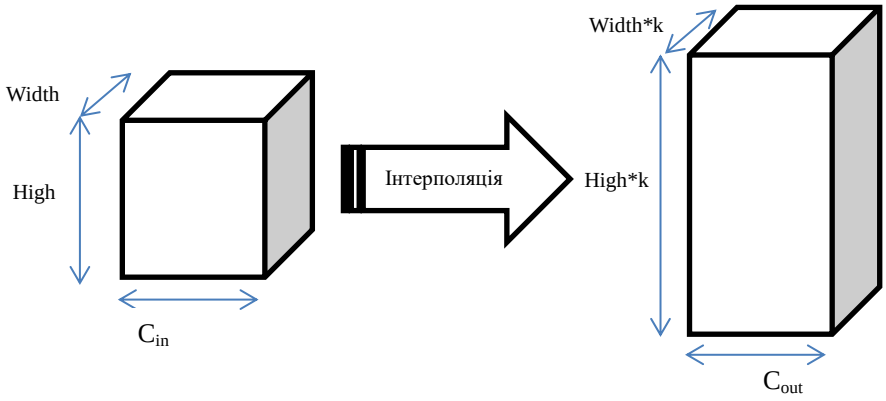


Рисунок 9 - Результат відновлення: High*Width формують площу каналу в шарі, C_{in} – вхідні канали, C_{out} – вихідні канали, k – коефіцієнт прорідження.

Архітектура ЗНМ полягає в тому, що кожен нейрон отримує вхідний сигнал з локального рецептивного поля у попередньому шарі. Це забезпечує локальну двовимірну зв'язність. За шарами згортки ідуть обчислювальні шари, які виконують певну фільтрацію (усереднення, максимізація) та створюють підвибірку, що забезпечує зменшення розширення для результату згортки (вихідного шару) [11].

Таким чином, кількість шарів НМ та кількість елементів в шарі визначає вимоги до необхідного об’єму пам’яті, а параметри ядер згортки та інтерполяційних ядер (функцій) до швидкодії, відповідно. З іншого боку обмежені ресурси пам’яті, обчислювальної потужності та часу на обробку змушують обмежувати кількість шарів, їх розміри, кількість каналів та розміри ядер згортки/інтерполяції.

Дана НМ використовує такі функції активації як сигмоїдальну, так і функцію випрямлення (relu) [13].

2.2 Особливості налаштування нейронної мережі для фільтрації артефактів (шумів) на цифрових зображеннях

Для знешумлення цифрових зображень застосовують спеціальну НМ – denoising autoencoder (DAE). Ця НМ використовує описаний раніше механізм згортки, проріджения і відновлення (формула 11, рисунки 6 – 9).

DAE складається з трьох частин: кодер, декодер і прихований простір (посередині). Структура DAE (рисунок 10) описана у джерелах [12], [13].

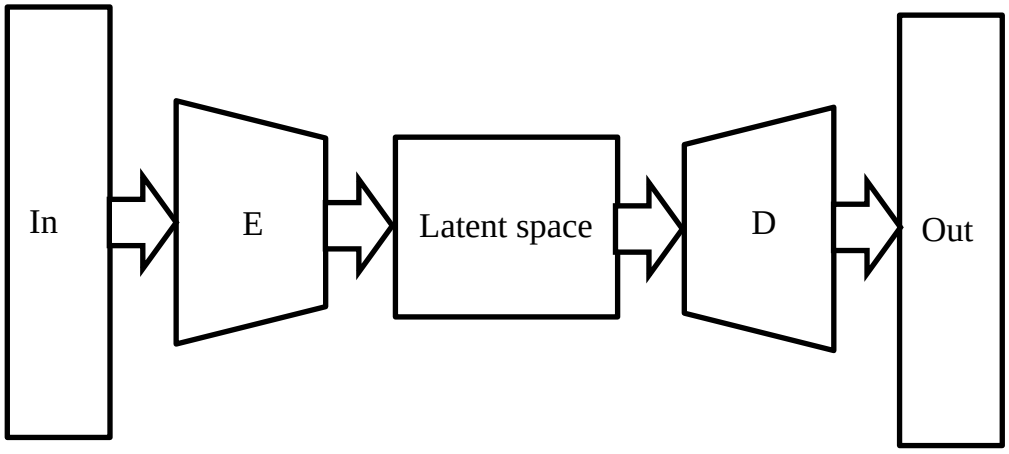


Рисунок 10 - Структура DAE (E – encoder/кодер, D – decoder/декодер, In – вхідний шар, Out – вихідний шар, Latent space – прихований простір/ шар ознак/ шар з великою кількістю каналів).

Кодер за допомогою згортки та проріджування перетворює вхідну інформацію відповідно вигляду прихованого простору. Декодер повертає дані, перетворюючи інформацію, подану у прихованому просторі, в такому ж вигляді, що отримує кодер.

Для проріджування було використано, найпростіший варіант з ядром згортки 2x2, при зменшенні кількісті елементів в шарі в 4 рази. Модель НМ складається з 7 шарів (за аналогією figure 2 у [14]). Частина НМ, в якій знаходиться кодер, використовує, окрім проріджування, для згортки ядра розміром 3x3 та додатковим відступом у 1 елемент (для компенсації крайових ефектів). При відновленні даних використовується ядро розміром 2x2 з кроком 2. Для переходу з представлення зображень в 3 каналах (RGB) було вирішено, що всі згорткові шари мають були однакового об'єму (C*H*W, де C – кількість каналів, H*W – площа каналу в шарі) і кратні 4, оскільк кожне проріджування зменшує кількість елементів в шарі вдвічі, тому створено перетворення з 3*H*W в 16*H*W/4.

Шари DAE (Д1):

1) Канали (3→16) – проріджування.
    a) Об'єм (3*H*W→16*H*W/4).
2) Канали (16→64) – проріджування.
    a) Об'єм (16*H*W/4→64*H*W/16).
3) Канали (64→256) – проріджування.
    a) Об'єм (64*H*W/16→256*H*W/64).
4) Канали (256→256) – відновлення.
    a) Об'єм (256*H*W/64→256*H*W/64).
5) Канали (256→64) – відновлення.
    a) Об'єм (256*H*W/64→64*H*W/16).
6) Канали (64→16) – відновлення.
    a) Об'єм (64*H*W/16→16*H*W/4).
7) Канали (16→3).

а) Об’єм (16*H*W.4→3*H*W).

2.3 Висновки за розділом

Таким чином, в цьому проєкті використано спеціальну ЗНМ, для знешумлення (DAE) цифрових зображень (RGB), з такими параметрами: 7 шарів згортки, однаковими об’ємами для внутрішніх шарів НМ, в якості операції проріджения використано фільтр максимуму (див. 2.2), ядра згортки в даному проєкті мають розміри 2x2 та 3x3.

РОЗДІЛ 3. ОПИС ПРОГРАМНОГО ЗАСОБУ

3.1 Загальна структура програмного засобу

Засіб розроблено за допомогою скриптової мови програмування Python з використанням бібліотек/фреймворків NumPy, openCV, pyTorch, matplotlib та pyforms.

NumPy — основний пакет наукових обчислень з Python [15]. Даний пакет широко використовується в даному проєкті, з митою швидкої обробки зображень. Для представлення даних часто використовується тип даних “ndarray” з цього пакету.

OpenCV (Open Source Computer Vision Library) — бібліотека програмного забезпечення з відкритим кодом та бібліотека програмного забезпечення для машинного навчання [16]. Зручна бібліотека для швидкої обробки даних.

PyTorch — система машинного навчання з відкритим кодом, що прискорює шлях від створення прототипу моделі досліджень до розгортання виробництва [17]. Відомі випадки помилок, викликані несумісністю фреймворків для розробки НМ з драйверами до відповідного обладнання, тому було використано pyTorch, оскільки таких помилок, не було виявлено. Для представлення даних, що взаємодіють з НМ використовується тип даних “tensor”.

Matplotlib - це широка бібліотека для створення статичних, анімованих та інтерактивних візуалізацій на Python [18]. В цьому проєкті використовується для побудови графіків навчання та тестування.

Pyforms — це фреймворк Python 3 для розробки додатків, здатних виконуватись як в настільному графічному інтерфейсі, так і в терміналі та веб середовищі [19]. Даний фреймворк простий у використанні, і розробка графічних інтерфейсів з його допомогою є популярною.

Засіб складається з 6 модулів, що пов’язані між собою (Д2).

3.2 Модуль обробки зображень (“images”)

Для обробки зображень за допомогою НМ необхідно мати достатню кількість внутрішньої пам’яті, щоб зчитати та обробити дані. У певних випадках, коли зображення займає кількість даних більшу, ніж дозволяє внутрішня пам’ять процесору, зображення необхідно сегментувати і обробити кожен сегмент окремо, після чого зображення треба відновити із оброблених сегментів (рисунок 11, Д3, Д4), щоб на стику не виникало зайвих артефактів (“ліній з’єднання”) можна використовувати певний відступ.

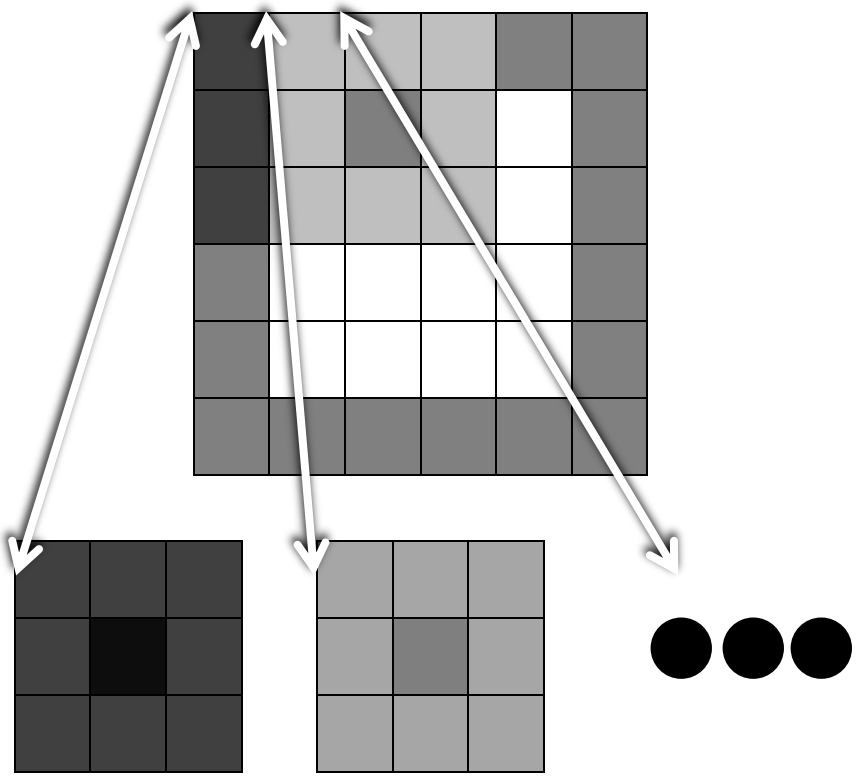


Рисунок 11 - сегментації/реконструкції.

Завданням цього модуля є обробка зображень за допомогою наступних функцій:

1. image_grid(shape_y, shape_x, squad):
   - приймає розміри основного зображення по осі y та x (shape_y, shape_x) і розміри фрагменту зображення квадратної форми (squad),

який повинен розділити основне зображення на шматки з відступами;

- повертає кількість квадратних зображень що вміщується в початковому зображенні, по осях x та y;

2. split_image(src_pth, squad, pad):
   - приймає рядок, що містить в собі шлях до зображення (src_pth), розміри фрагменту зображення квадратної форми (squad), та розміри відступу (pad);
   - пробує зчитати зображення, якщо не вдається то повертає False, 3xNone та рядок з інформацією про помилку, інакше — повертає True, нарізані зображення у вигляді масиву ndarray, розміри початкового зображення, і None замість рядка про помилки;
3. split_image_to_file(dst_path_dir, file_name, array):
   - приймає рядок, що містить шлях до теки призначення (dst_path_dir), рядок зі спільним ім’ям файлів призначення (file_name) та масив нарізаних зображень (array);
   - пробує записати зображення додаючи до імені числа, відповідно до розташування у масиві, у разі невдачі;
   - повертає False та рядок з інформацією про помилку, інакше — True та None;
4. split_images_to_dir(src_pth, dst_pth, squad, pad, lst, progress):
   - приймає рядок, що містить шлях до теки з зображеннями користувача (src_pth), рядок, що містить шлях до теки в яку треба зберігати оброблені зображення (dst_pth), розмір фрагменту зображення квадратної форми (squad), відступ (pad) та вказівник на індикатор виконаних завдань (progress);
   - за допомогою split_image_to_file записує зображення в теки призначення;

- якщо передано progress, то значення його змінної збільшувати в процесі обробки;
- повертає значення True, якщо всі дані записані коректно, і False — інакше, та відповідне значення рядка помилки отримане з split_image_to_file;

5. marge_image(shape_y, shape_x, squad, pad, array):
   - приймає розміри кінцевого зображення по осі y та x (shape_y, shape_x), розміри фрагменту зображення квадратної форми (squad), відступ (pad), та масив зображень (array);
   - повертає зклеєне зображення кінцевого розміру в форматі ndarray;
6. split_range(lst, sp):
   - приймає список з файлів (lst), та кількість відсотків для розподілу на дані для тренування і дані для тестування (sp);
   - повертає два нові списки: список файлів, що будуть використовуватися для тренування і список — для тестування;
7. check_images_shapes(tr, te, trv, tev, l_tr, l_te, l_trv, l_tev, progress):
   - приймає рядки, що містять шляхи до тек з підготовленими зображеннями (tr — дані для тренування, te — для тестування, trv — для перевірок тренування, tev — для перевірок тестування), списки з іменами файлів (l_*) та індикатор виконаних завдань (progress);
   - якщо передано progress, то значення його змінної збільшувати в процесі обробки;
   - повертає значення True, якщо всі файли мають однаковий розмір, інакше — False, та розмір першого зображення.
8. plot_one(x, y, x_label, y_label, x_scale, y_scale):
   - приймає послідовності x, y, назви осей (x_label, y_label), та масштаби осей (x_scale, y_scale);
   - повертає зображення графіку у вигляді масиву;
9. plot_two(x, y1, y2, x_label, y_label, x_scale, y_scale):

- приймає послідовності x, y1, y2, назви осей (x_label, y_label), та масштаби осей (x_scale, y_scale);
- повертає зображення графіку у вигляді масиву, та місце перетину графів;

10.get_image(*args, **kwargs):

- викликає функцію бібліотеки openCV imread і передає в неї змінні (*args, **kwargs);

11.set_image(*args, **kwargs):

- викликає функцію бібліотеки matplotlib imsave і передає в неї змінні (*args, **kwargs);

3.3 Модуль створення дерева файлів (“directorys”)

Оскільки бібліотека PyTorch може використовувати картинки користувача, то підготовлені зображення необхідно зберігати в теці певного дерева файлів (рисунок 12).

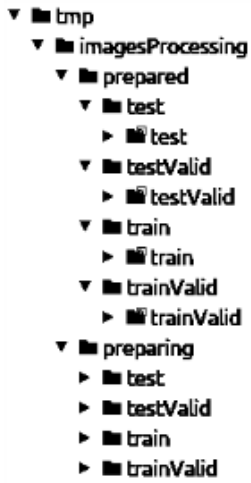


Рисунок 12 - Дерево файлів.

Цей модуль містить наступні функції для створення такого дерева:

1. if_files_exist(pth, ftr):
   - приймає рядок, що містить шлях до теки (pth) та рядок (ftr, “фільтр”), в якому вказується формат шуканих файлів;
   - повертає список файлів, та повідомлення про помилку (або None).
2. make_dir(pth):
   - приймає рядок, що містить шлях до теки (pth);

- пробує створити теку, або відмітити, що вона вже існує;
- повертає True у разі успіху, інакше — False.

3. make_symlink(src, dst):
   - приймає рядок, що містять шляхи джерела та призначення тек (src, dst);
   - пробує створити символьне посилання;
   - повертає True у разі успіху, інакше — False.
4. linking_prepared_data(tr, te, trv, tev):
   - приймає рядки, в яких міститься шляхи до тек із заздалегідь підготовленими даними (tr — дані для тренування, te — для тестування, trv — для перевірки тренування, tev — для перевірки тестування);
   - створює теки за допомогою make_dir, та посилання для них, використовуючи make_symlink;
   - повертає True у разі успіху, інакше — False.
5. create_dir_for_new_data(pth):
   - приймає рядок, що містить шлях до теки (pth);
   - якщо такий шлях існує, то видаляє його;
   - створює теку;
   - повертає True у разі успіху, інакше — False;
6. linking_preparing_data():
   - створює теки (create_dir_for_new_data) для підготовки даних та символьні посилання на них (linking_prepared_data);
   - повертає True у разі успіху, інакше — False;
7. unlink_dir(pth):
   - приймає рядок, що містить шлях до теки (pth);
   - пробує видалити посилання за цим шляхом;
   - повертає True у разі успіху, інакше — False;
8. unlink_prepared_data():

- за допомогою unlink_dir, видаляє символьні посилання на теки з підготовленими даними;
- повертає True у разі успіху, інакше — False;

9. remove_prep_data():
   - пробує видалити всі дерево файлів в якому знаходяться підготовлені дані;
   - повертає True у разі успіху, інакше — False.

3.4 Модуль нейронної мережі (“nnmodel”)

В цьому модулі знаходиться сама НМ, та функції для роботи з нею. НМ представлена за допомогою класу, що успадковує клас бібліотеки pyTorch, і містить наступні складові:

- три шари згортки для кодера;
- середній шар згортки (розгортки);
- три шари розгортки для декодера;
- перевизначену функція опису НМ (forward);
- функцію запису значень ваги коефіцієнтів НМ у файл (save_model);
- функцію зчитування ваги коефіцієнтів НМ з файлу (load_model);

Також для роботи з цією НМ, модуль містить наступні функції:

1. create_data_loader(pth, batch_size):
   - приймає рядок, що містить шлях до теки (pth), в якій знаходиться тека з зображеннями та кількість зображень, що може бути зчитана за один раз (batch_size);
   - повертає об’єкт-завантажувач зображень;
2. train_validation(batch_size):
   - приймає кількість зображень, що може бути зчитана за один раз (batch_size);
   - за допомогою, заздалегідь визначених тек, створює завантажувачі зображень для тренування;

- повертає завантажувач зображень тренування, і завантажувач зображень перевірки.

3. test_validation(batch_size):
   - приймає кількість зображень, що може бути зчитана за один раз (batch_size);
   - за допомогою, заздалегідь визначених тек, створює завантажувачі зображень для тестування;
   - повертає завантажувач зображень тестування, і завантажувач зображень перевірки.
4. testing_losses(model, progress):
   - приймає модель НМ (model) і вказівник на індикатор виконаних завдань (progress);
   - використовуючи test_validation (batch_size = 1), за допомогою циклу, тестуються тестові зображення після обробки НМ відносно зображень перевірки;
   - якщо передано progress, то значення його змінної збільшувати в процесі обробки;
   - повертається список втрат після тестувань, визначений за допомогою середньо квадратичного відхилення, та час тестування всіх зображень (в секундах);
5. test_part_image(model, t_image):
   - приймає модель НМ (model) і зображення у форматі tensor (t_image);
   - повертає, оброблене за допомогою моделі, зображення у вигляді масиву формату ndarray NumPy;
6. split_tensor(tens_pth, squad, pad):
   - аналогічна до функції split_image з модуля images.py, використовує функцію get_image;
   - повертає зображення у форматі tensor;
7. test_image(model, testing_image_pth, squad, pad, progress):

- приймає модель НМ, рядок, що містить шлях до тестового зображення, розміри фрагменту зображення квадратної форми, розмір відступу та вказівник на індикатор виконаних завдань (progress);
- за допомогою split_tensor отримує зображення, тестує всі зображення з масиву, після чого, з використанням marge_image з модуля images.py зклеює вихідне зображення;
- якщо передано progress, то значення його змінної збільшувати в процесі обробки;
- в разі успіху повертає True, зображення в форматі ndarray, час виконання обробки з НМ, і None, інакше — False, 2xNone, рядок з повідомленням про помилку;

8. train(mod, batch_size, epochs, progress):
   - приймає модель НМ (mod), кількість зображень, що може бути зчитана за один раз (batch_size), кількість епох тренування (epochs) і вказівник на індикатор виконаних завдвнь;
   - використовуючи train_validation (batch_size = batch_size), за допомогою циклу, по тренувальних і перевірочних зображеннях (з кількістю зчитаних даних – batch_size), виконується тренування НМ;
   - якщо передано progress, то значення його змінної збільшувати в процесі тренування;
   - повертає модель НМ, список з втрат (обчислених за допомогою середньо-квадратичного відхилення, після кожної епохи) і час тренування (в секундах).

3.5 Модулі графічного інтерфейсу

Графічний (користувацький) інтерфейс реалізовано такими модулями:

- main;
- preparing;

- testing.

“Preparing ” – реалізує частину інтерфейсу, призначену для підготовки зображень до обробки. “Testing ” – реалізує частину інтерфейсу, що відповідає за взаємодію з НМ (тренування, збереження моделі та тестових зображень, тестування НМ, завантаження ваги НМ з файлу). “Main” - точка входу при виконанні скрипту інтерпритатором і зв’язує “preparing” з “testing”. Всі ці модулі використовують фреймворк pyforms.

Для реалізації інтерфейсу використані такі компоненти:

1) ControlButton – об’єкт, що починає виконувати функцію, вказівник якої знаходиться в змінній “value” цього об’єкта.
2) ControlCheckBox – за допомогою цього об’єкту, користувач визначає одне з двох значень, що може містити змінна його (“value”);
3) ControlDir – цей об’єкт реалізує введення даних з клавіатури, або з допомогою “файлового провідника” (шлях до теки), під час обробки введеного рядка (змінна “value”) перевизначеною функцією (“changed_event”), аналізує “value”, якщо дані введені коритувачем не правильні, то очищує змінну “value” і виводить повідомлення про помилку з використанням відповідного ControlLabel;
4) ControlFile – виконує ті ж функції, що і ControlDir, в даному проєкті використано з метою завантаження тестових зображень та файлу, який містить вагу коефіцієнтів НМ.
5) ControlLabel – цей об’єкт реалізує виводи повідомлень для користувача;
6) ControlNumber – за допомогою цього об’єкта, користувач може ввести числа, що будуть використані в програмі.
7) ControlProgress – об’єкт (індикатор кількості виконаних задач), що показує користувачу стан виконання процесу від 0 до 100 відсотків.

8) ControlText – об’єкт, що дозволяє користувачу ввести текст з клавіатури, в даному проєкті його використано для введення імені файлів, які буде збережено, у разі не правильно введеної назви, поле введення буде очищено.

Повідомлення про виняткові ситуації, які виводяться користовачу:

- файл – не існує (“Exist exception:…”);
- тека вже існує (“already exists:…”);
- теку було створено (“dir … has been created”);
- символьне посилання на теку не створено (“make symlink exception”);
- видалення дерева файлів (“removing … exception”);
- видалення символьного посилання (“unlink dir … exception …”);
- при читанні зображення (функція в програмі яка читала);
- при запису зображення (функція в програмі яка записувала);
- не правильні назви зображень (не збігаються) для обробки, або їх відсутність (“wrong empty ranges or the name do not match”);
- не правильна тека, або розміри зображень (“Dir or shape err…”);
- виняткова ситуація під час тренування НМ.

Приклади роботи графічного інтерфейсу розробленого програмного засобу зображено на рисунках 13-15.

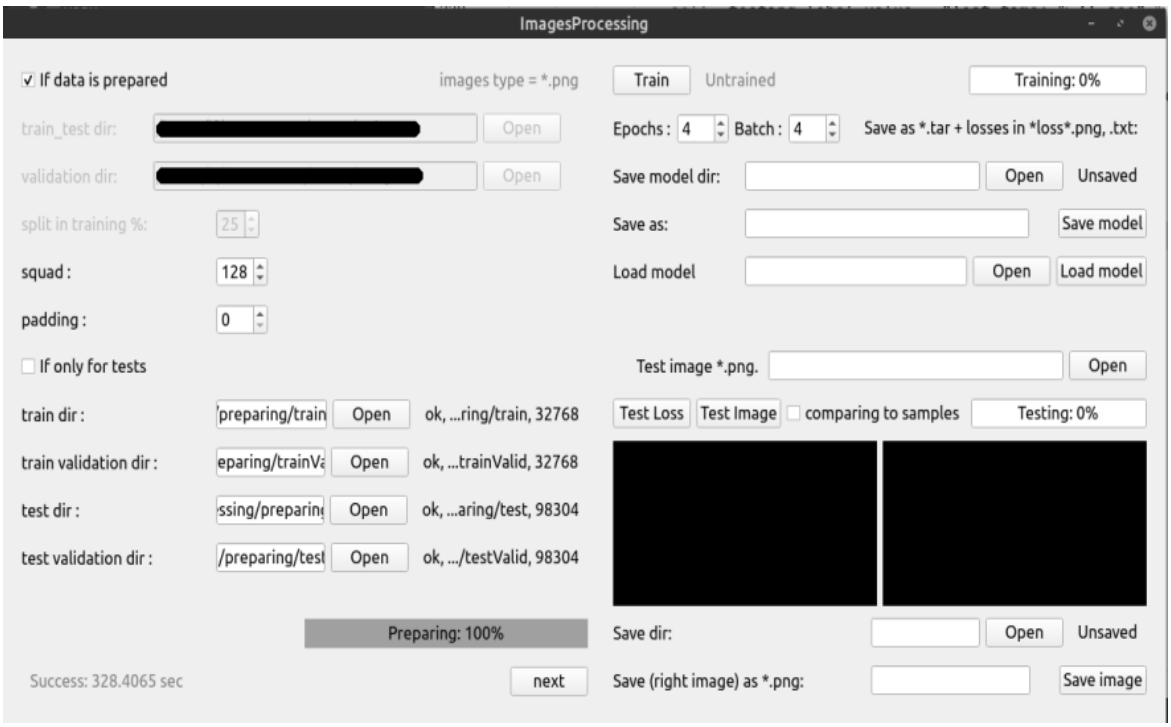


Рисунок 13 - Приклад розробленого графічного інтерфейсу (1).

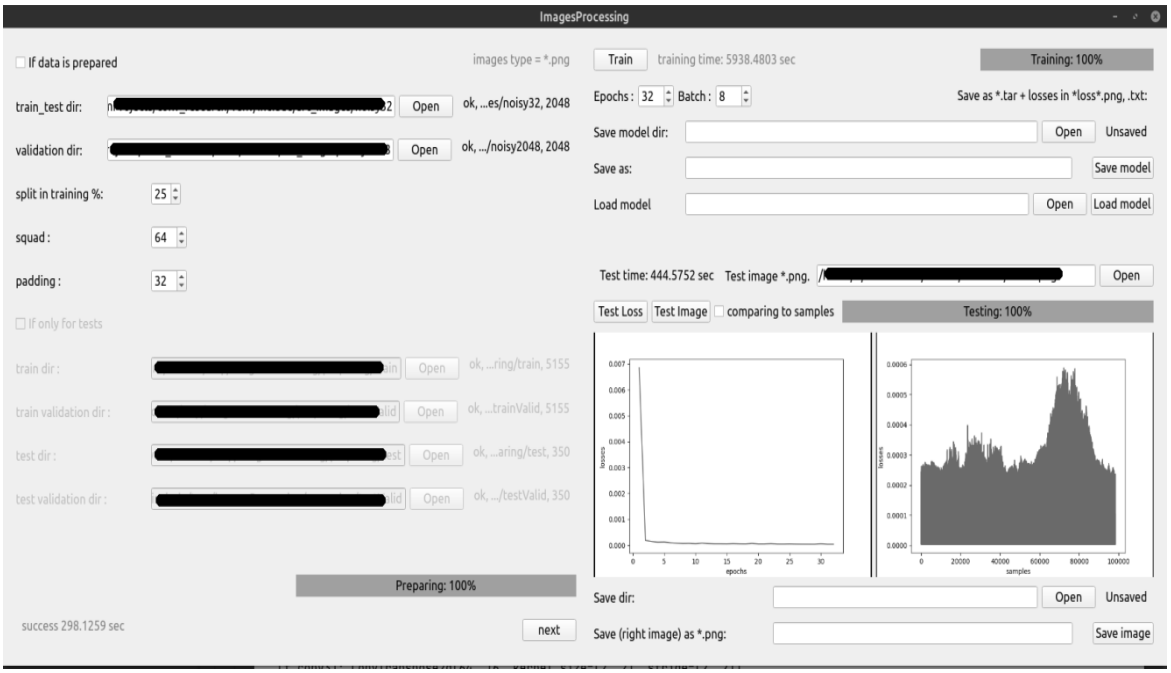


Рисунок 14 - Приклад розробленого графічного інтерфейсу (2).

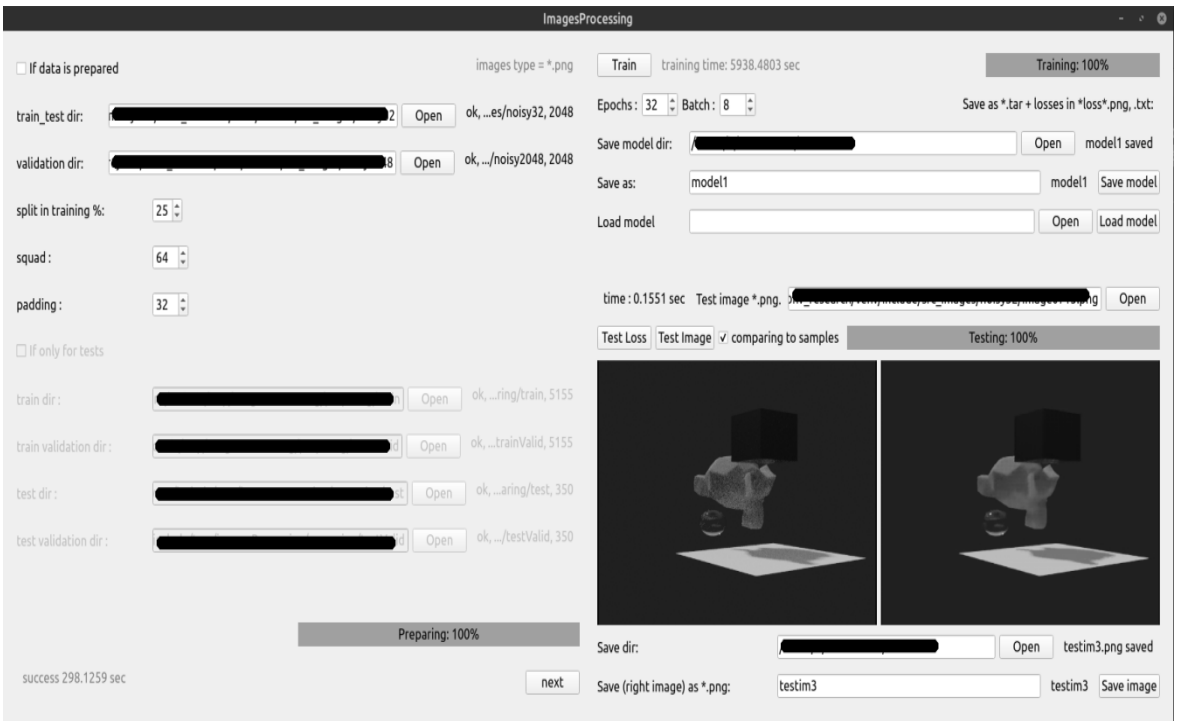


Рисунок 15 - Приклад розробленого графічного інтерфейсу (3).

3.6 Висновки за розділ

Програмний засіб складається з 6 модулів. Модулі графічного інтерфейсу пов’язані між собою, та використовуються для задання параметрів підготовки даних, вказування джерел даних, тренування і тестування НМ. В разі виникнення виняткових ситуацій, програма продовжує виконуватися та дозволяє створювати (дотреновувати) декілька моделей впродовж виконання.

РОЗДІЛ 4. ПЕРЕВІРКА РІШЕННЯ

4.1 Показники якості обробки і характеристика наборів зображень

Тренування (тестування) НМ відбувається шляхом обробки тренувальних зображень та оцінювання (порівняння) результатів цієї обробки з очікуваними зображеннями (Д5, Д6). Середньоквадратичне відхилення виступає в ролі способу оцінювання:

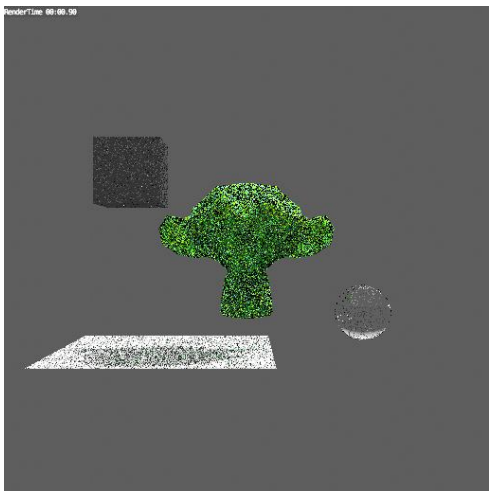
$$\sigma^2 = \frac{1}{N}\sum_{i=0}^{N}(X[i] - X_0[i])^2, \quad (12)$$

де N – кількість елементів масиву, який представляє зображення, X - елемент масиву зображення, що обробляється, X_0 – елемент масиву очікуваного зображення, і та j – індекси (координати) пікселів.

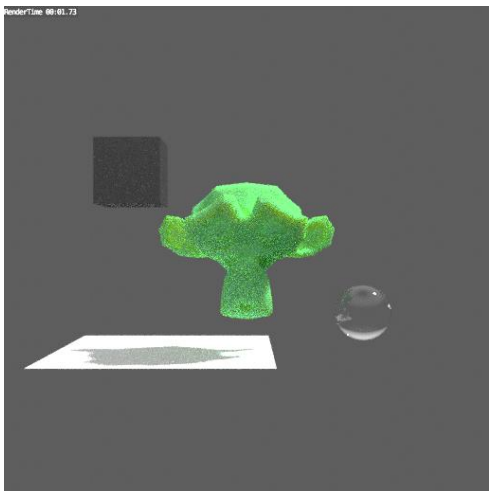
В якості очікуваних зображень було обрано 2048 “пласких” проєкцій тривимірної сцени з кількістю вибірок 2048 (загальний час рендерингу всіх зображень зайняв 29,065 год., середній час для одного зображення складає 51.09 сек.). Зображення, що були використані для тренування (або тестування) мають наступні характеристики:

- 2048 “пласких” проєкцій з тривимірної сцени з однією вибіркою, та проєкції нормалей від площин тривимірних об’єктів сцени, де колір залежить від куту нормалі до проєкції, а також проєкції базових кольорів цих об’єктів (тобто тут створено 3 типи зображень для обробки з допомогою НМ, загальний час рендерингу зайняв 0.4778 год, середній час рендерингу одного зображення склав 0.84 сек.).
- 2048 “пласких” проєкцій з тривимірної сцени з кількістю вибірок 32 (загальний час рендерингу всіх зображень зайняв 0.8988 год., середній час для одного зображення складає 1.58 сек.).

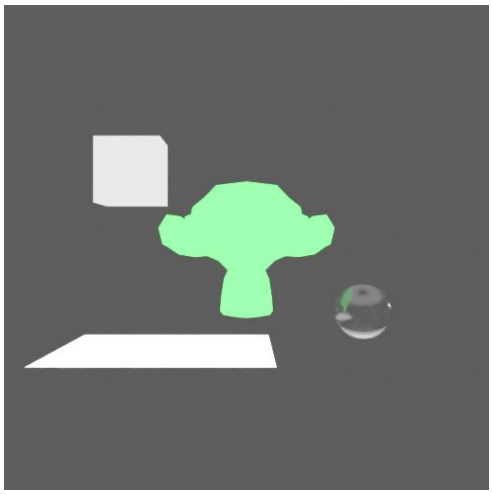
Ці зображення представлені рисунком 16.



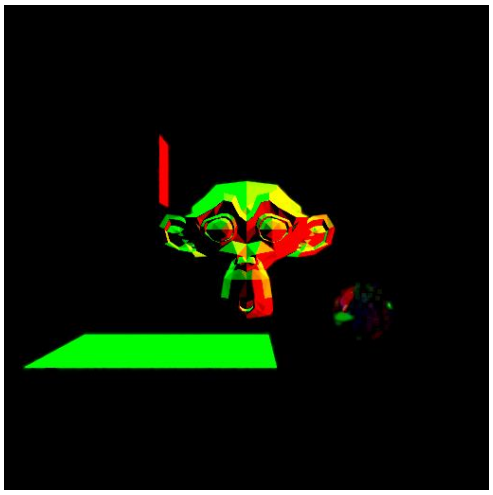
а



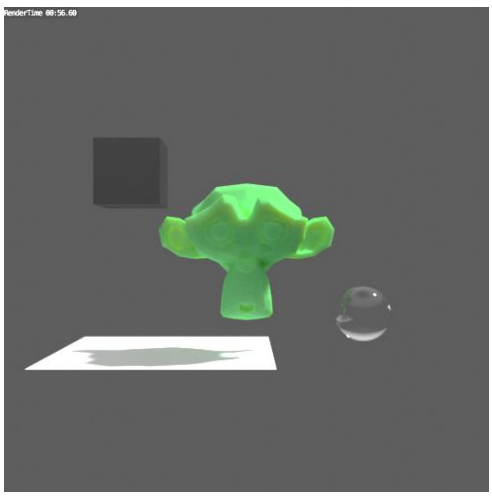
б



в



г



Д

Рисунок 16 - Типи зображень створені за допомогою рендерингу (а – 1 вибірка, б – 32 вибірки, в – базовий колір об’єкта, в – “нормалі”, д – очікуване зображення, 2048 вибірок).

Для обчислення залежності між кількістю вибірок і рівнем втрат на зображенні (середньоквадратичне відхилення) було створено набір зображень (256 “пласких” проєкцій), де кількість вибірок варіюється від 1 до 256 (послідовно). Оскільки рендеринг будь яких сцен займає багато часу то було вирішено обмежитися такою кількістю зображень, також тести тренування НМ проходили не довго, що не дозволяло досягти рівня втрат меншого ніж у зображень з 256 вибірками. В майбутньому, для оцінювання втрат, ця кількість зображень має бути розширена, щоб покрити весь проміжок зображень з різними вибірками, аж до очікуваного зображення.

4.2 Порядок проведення перевірки

Для проведення тестів необхідно встановити:

- розподіл зображень на тренувальні і тестові;
- кількість епох тренування;
- batch size (кількість зображень, що може оброблятися одночасно).

Якщо зображення завеликі для пам’яті системи, то їх необхідно сегментувати. Після чого підготовлені зображення зберігаються у відповідному файловому дереві, з якого, за допомогою завантажувачів можна зчитати їх і використовувати в процесі навчання, або тестування НМ.

Після одержання даних тренування НМ, збережено дані про тренування у вигляді графіків залежностей між епохами та середньо-квадратичним відхиленням у різних масштабах.

Після цього оброблено зображення з тестової частини і представлено результати у вигляді графіків, де порівнюються рівні втрат зображень згенерованих рендерингом і оброблених НМ зображень.

Кінцевим етапом, можна перевірити вигляд оброблених НМ зображень і переконатися, що не з’явилися нові артефакти.

4.3 Результати перевірки

Залежності між втратами і кількістю вибірок і часом рендерингу зображені на рисунку 17.

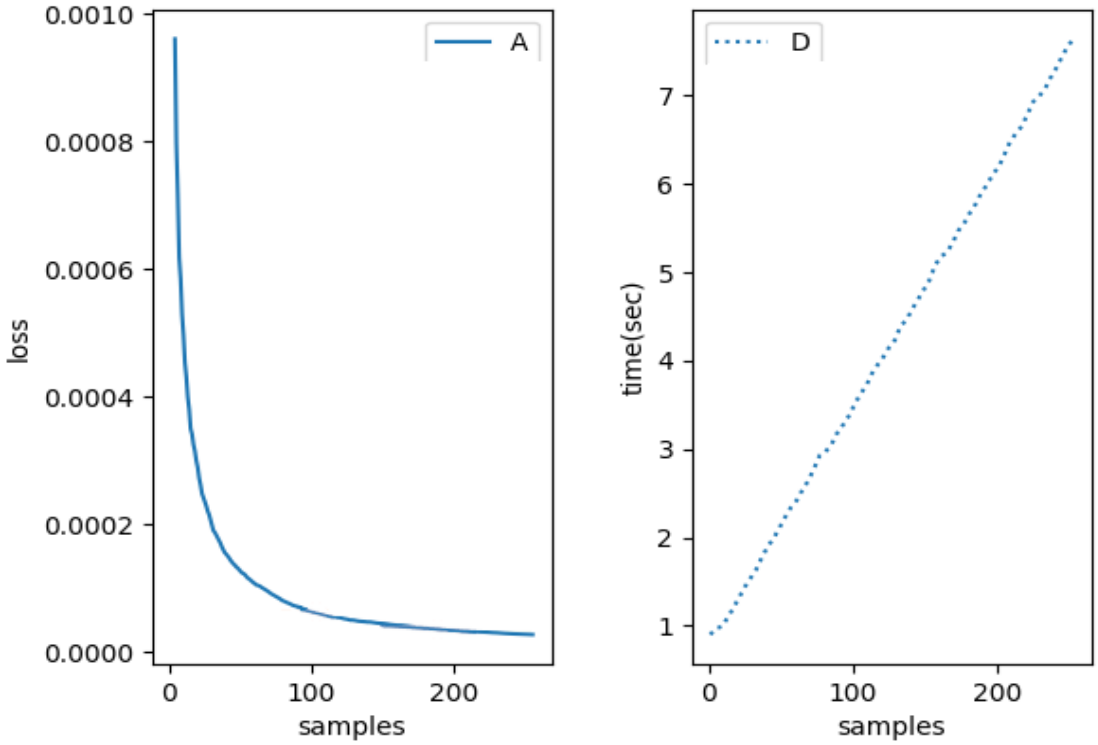


Рисунок 17 - Залежності між вибірками, втратами (середньоквадратичне відхилення) і часом (в секундах) рендерингу.

Графік залежності втрат від вибірок (Рисунок 17, “A”) має логарифмічну залежність – зі збільшенням кількості вибірок, швидкість збільшення якості зображення спадає, проте час рендерингу залежить від кількості вибірок лінійно – графік (Рисунок 16, “D”).

Спочатку було перевірено як впливає розподіл на тренувальні і тестові зображення на якість тренування. Ці результати представлені на рисунку 18.

Параметри зображень і НМ:

- НМ тренувалася знешумлювати зображення з 32 вибірками;

- Кількість епох тренування – 64;
- Кількість даних, що може бути зчитана за раз – 4;
- Розміри зображень – 512x512 (без сегментації та відступів).

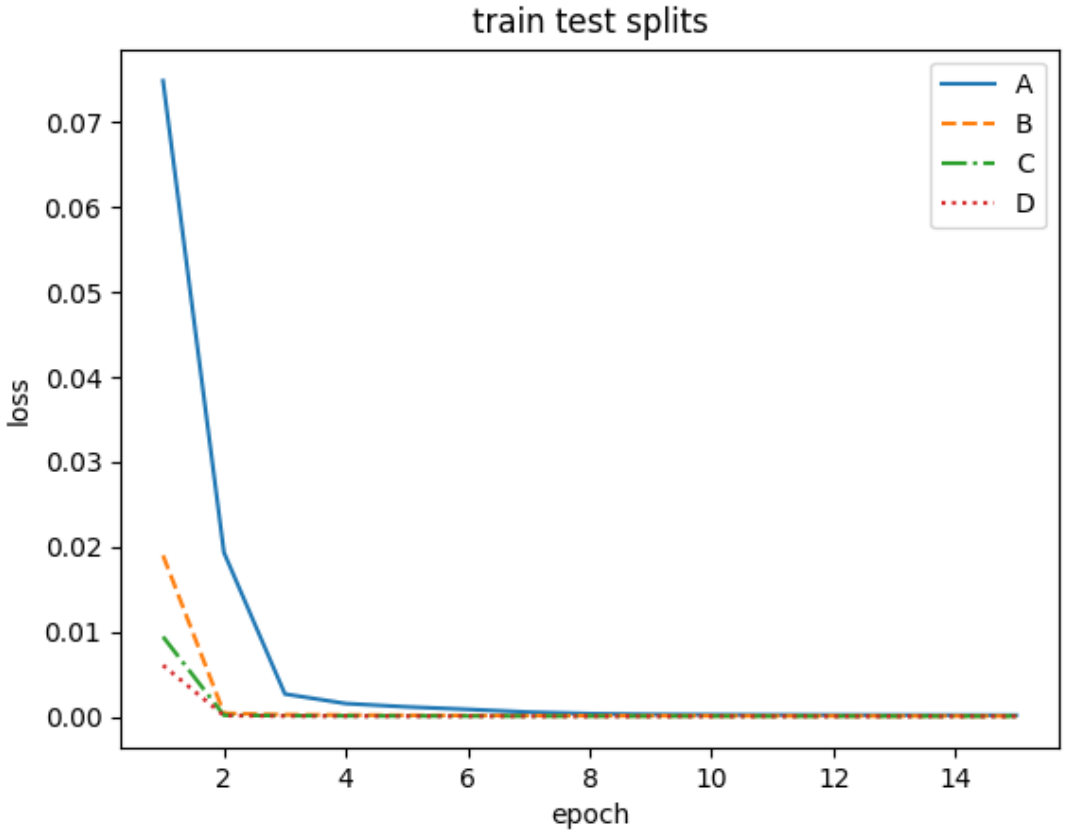


Рисунок 18 - залежність втрат від розподілу (під час тренування), де “A” – 103 зоб. (5%), “B” – 512 зоб. (25%), “C” – 1024 зоб. (50%) , “D” – 1536 зоб. (75 %).

З графіку, зображеному на рисунку 18, видно, що чим більше даних НМ отримає під час навчання, то меншу кількість епох необхідно встановити. Графік зображає процес навчання на відрізку від 1 до 15 епох, оскільки суттєвої різниці (1e-4 – 1e-6) не видно вже після 8 епох. Оскільки алгоритм навчання проходить одні і ті самі дії, то час навчання залежить від кількості зображень, що використовуються в навчанні, а не від кількості епох. Надалі буде використовуватися набір зображень для тренування з 103 екземплярів (5% від 2048), з метою економії часу тренування.

Для перевірки залежності часу було взято масив зображень з 32 вибірками. На основі результатів тренування було побудовано графік, що зображений на рисунку 19. Для виводу часу обробки за одну ітерацію циклу, було тимчасово модифіковано функцію тренування, шляхом додавання виводу цієї інформації в термінал. Після чого дані були оброблені за допомогою MatPlotLib (див. 3).

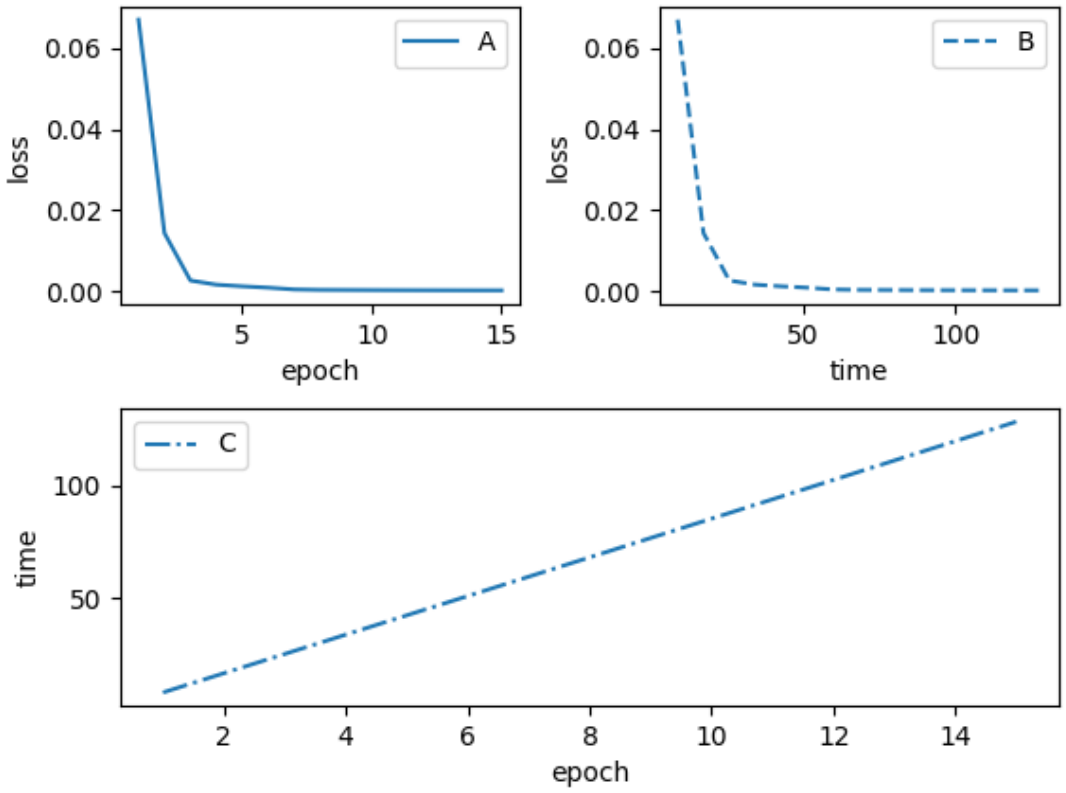


Рисунок 19 - Залежності втрат, епох та часу (в секундах) в процесі тренування.

Графік “C” показує, що час тренування залежить від кількості епох (в одній епосі 103 зображення) лінійно, тому графіки “A” та “B” виглядають однаково.

Наступною перевіркою було обрано зображення різних типів, які були описані в 4.1 та встановлено по 64 епохи тренування з кожним типом зображення. Результати цієї перевірки представлені на рисунку 20.

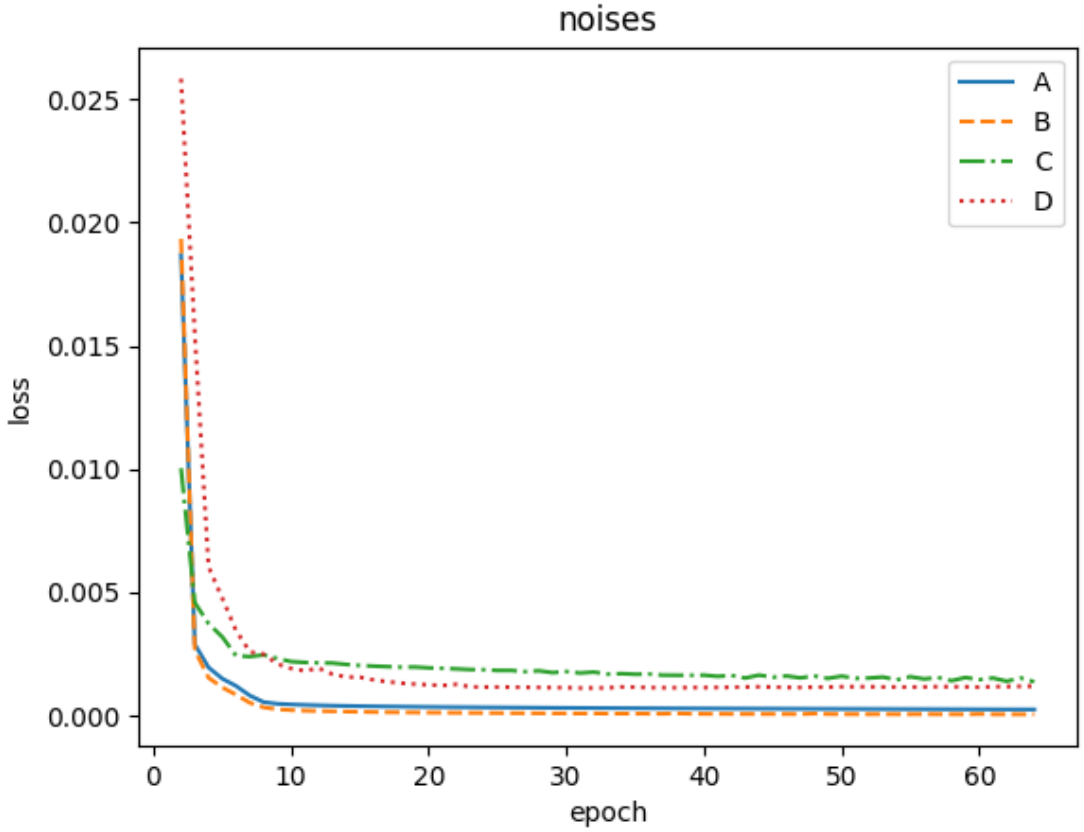


Рисунок 20 - Залежність втрат від типу зображення в процесі тренування (A – 1 вибірка, B – 32 вибірки, C – базовий колір об’єктів, D - “нормалі”).

Як видно з рисунку 20, зображення з артефактами обробляються краще (“A”, “B”), ніж інші типи зображень. Варто зазначити, що дана НМ може трансформувати одне представлення зображення в інше, але втрати будуть помітними.

Для того щоб оцінити якість результатів після фільтрації необхідно використати залежність між кількістю вибірок і середньоквадратичним відхиленням, що була описана в 4.1, після чого визначити точку перетину між графіком залежності втрат від вибірок і рівнем втрат на тестовому наборі зображень. Розглянемо два випадки, коли НМ тренується з набором зображень (512x512, без сегментації та відступів) з 32 вибірками та різною кількістю епох. Результати представлено в таблиці 1.

Таблиця 1 - Порівняння тренувань з різною кількістю епох.

Кількість епох	Час тренування	Еквівалент рівня втрат у вибірках при рендерингу
64	557.00 сек.	108
256	2185.99 сек.	186

Для порівняння результату рендерингу зображень з 32 вибірками і фільтрації за допомогою НМ зі звичайним рендерингом зображень 186 вибірками, з таким же рівнем середньоквадратичним відхиленням, вирішено таку систему рівнянь:

$$\begin{cases} 32 \text{ вибірок} = k * 1.58 \text{ сек./зоб.} + b; \\ 2048 \text{ вибірок} = k * 51.09 \text{ сек./зоб.} + b; \\ 186 \text{ вибірок} = k * x + b, \end{cases} \quad (13)$$

де k, x, і b – змінні, 1.58 та 51.09 – середній час рендерингу одного зображення.

$$\begin{cases} k = 40.72; \\ x = 5.36 \text{ сек./зоб.} \end{cases}. \quad (14)$$

Оскільки залежність вибірок від часу лінійна, ми можемо вирахувати приблизний час рендерингу зображення з відповідною кількістю вибірок.

Для рендерингу 2048 зображень з 186 вибірками нам необхідно близько 3.049 год. (10977.28 сек.). Щоб обробити всі 2048 зображень за допомогою натренованої НМ необхідно 88.72 сек. Щоб натренувати мережу для знешумлення до такого рівня необхідно 1.5 год. (5425.01 сек.) на рендеринг зображень, 0.6 год. (2185.99 сек.) на тренування НМ і 84.26 сек. – на обробку решти зображень. Якщо враховувати час тренування НМ, то виграш у часі становить близько 1.4264 рази, якщо використовувати уже натреновану НМ – 123.729 рази.

Щоб перевірити якість зображень в разі їх обробки із сегментацією було зроблено 4 тести. Параметри цих тестів знаходяться в таблиці 2.

Таблиця 2 - Порівняння тестів із сегментацією зображень.

Тест	1	2	3	4
Розмір зоб. (піклелі)	512x512	512x512	512x512	512x512
Кількість зображень	103	103	52	52
Відступи (пікселі)	0	56	0	24
Розмір внутрішнього сегменту (пікселі)	256x256	256x256	192x192	192x192
Час тренування	70 сек.	132 сек.	46 сек.	77 сек.
Кількість епох	10	10	10	10

Спосіб задання параметрів для тренування і проведення тестів показано на рисунках 21-24. Оскільки тренування – це довгий процес, то для 3 і 4 тесту було взято 80% з 65 (52) зображень для тренування і решту для тестів (тієї ж тривимірної сцени, що і для перших двох тестів, де кількість зображень дорівнює 103, або 5% від 2048). Цієї кількості зображень і епох тренування достатньо для того, щоб показати, як сегментація впливає на крайові ефекти в кожному з сегментів, але не достатньо, щоб відфільтроване зображення було чітким.

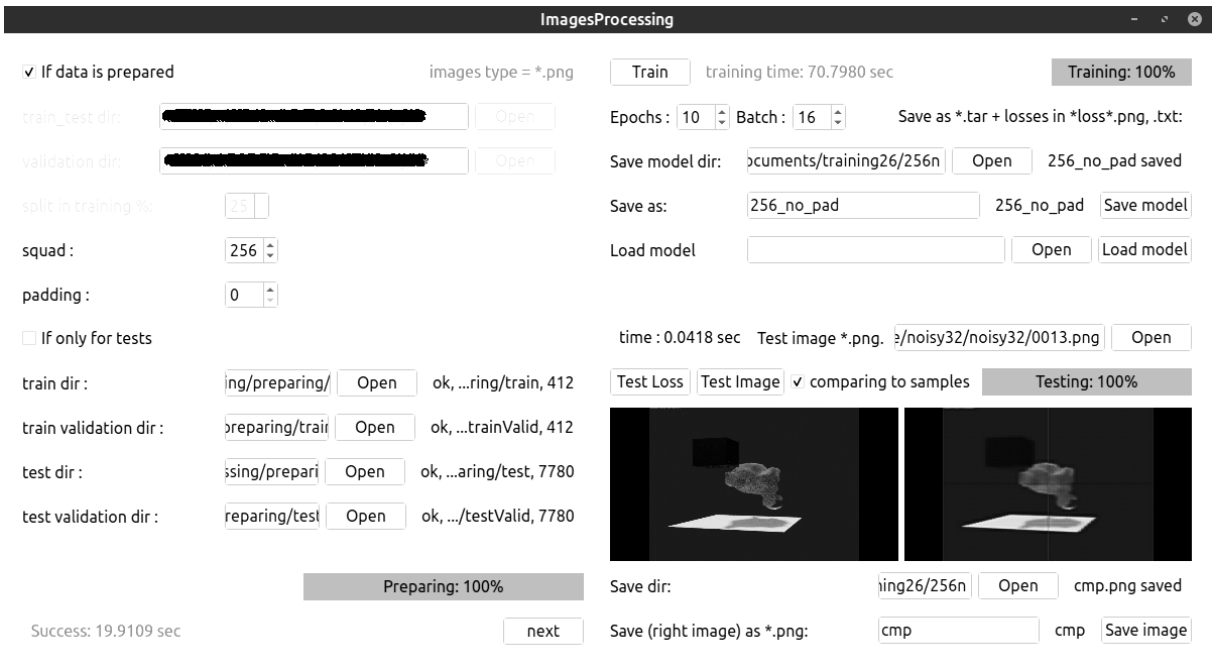


Рисунок 21 - Задання параметрів для 1 тесту.

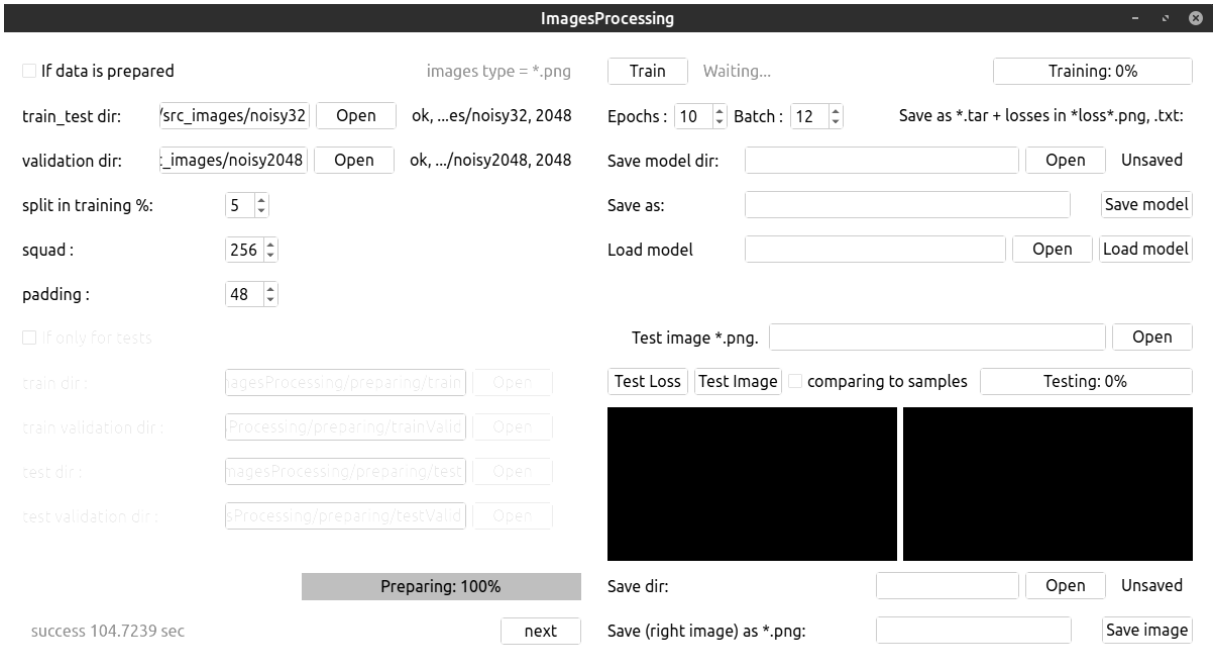


Рисунок 22 - Задання параметрів для 2 тесту.

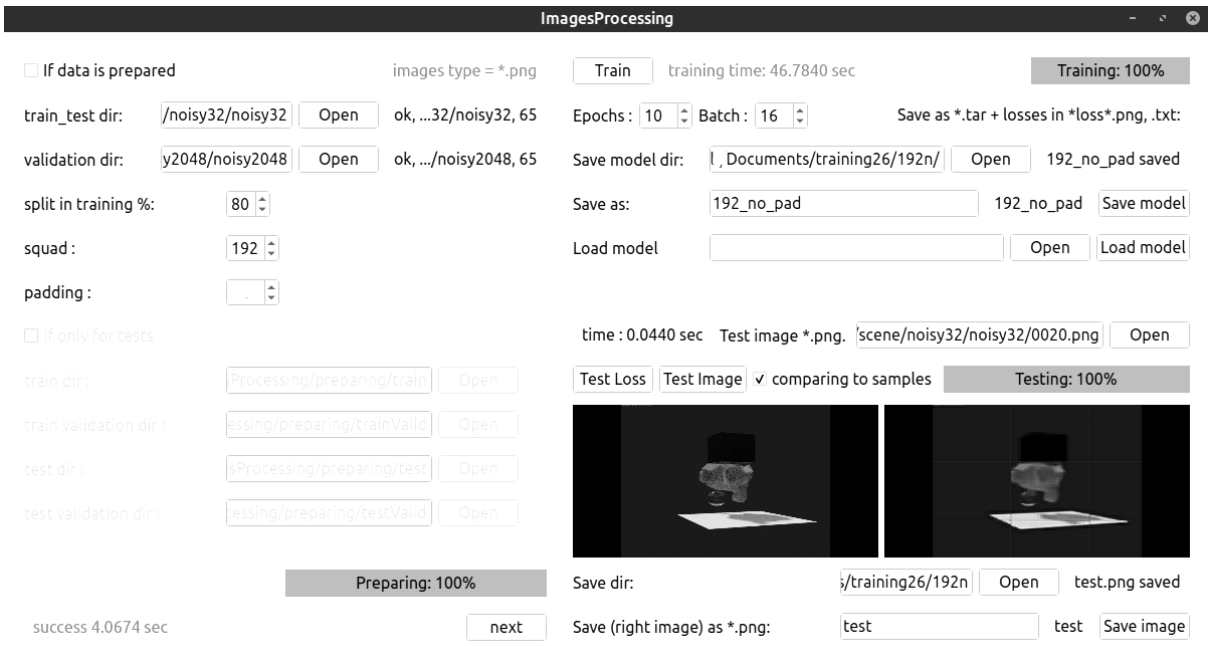


Рисунок 23 - Задання параметрів для 3 тесту.

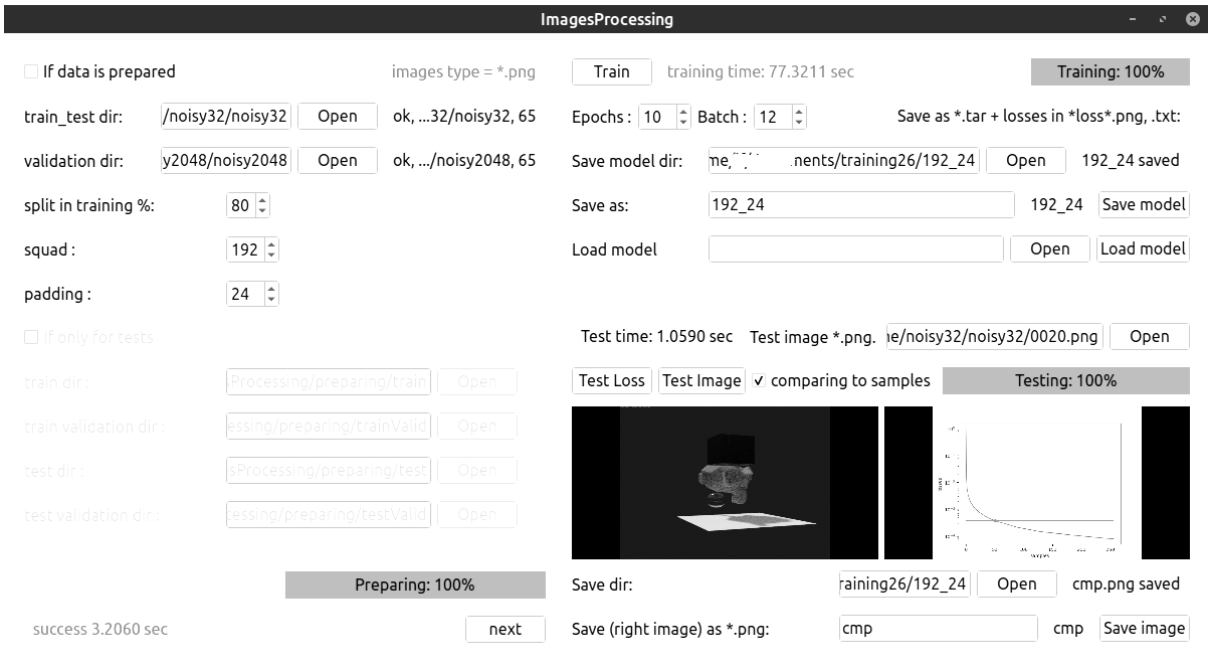
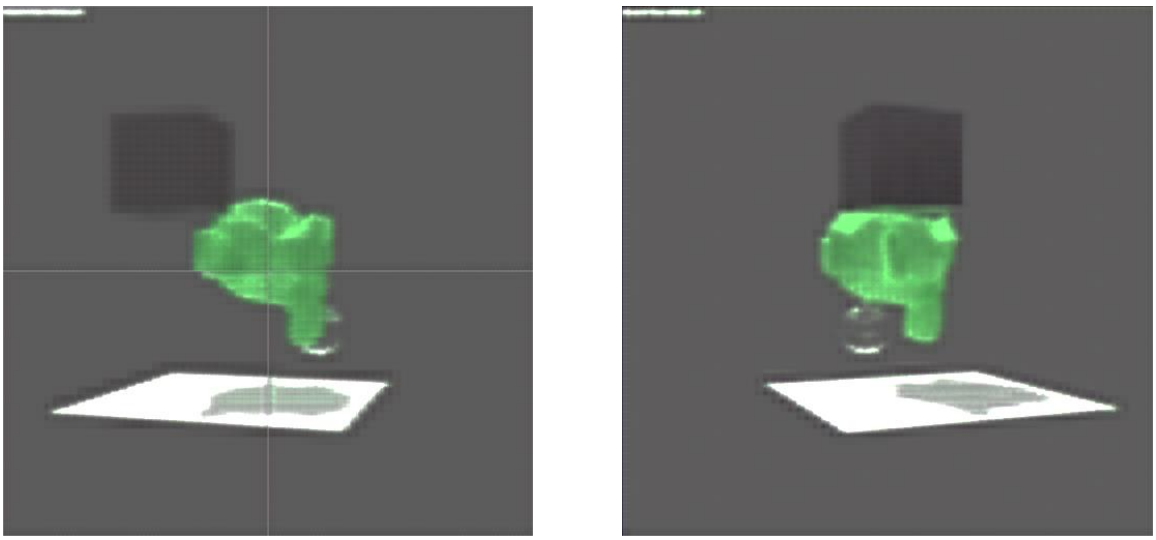


Рисунок 24 - Задання параметрів для 4 тесту.

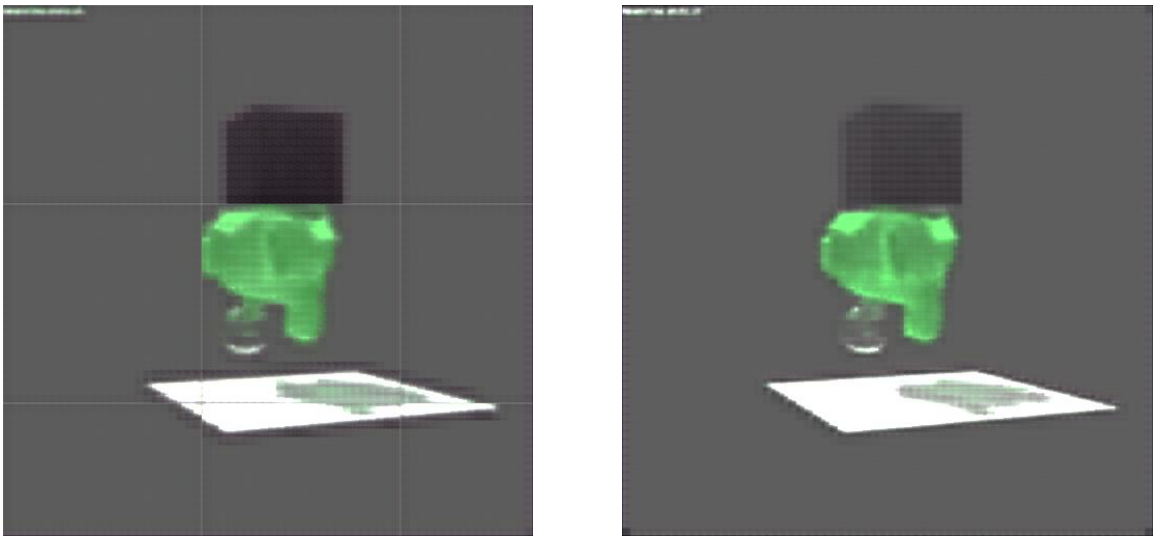
На рисунках 25-26 показано відновлені зображення з відступами і без з різними розмірами сегментів. Тестові зображення було взято випадково з теки, де вони знаходилися.



А Б

Рисунок 25 - результати тестів: а - 1, б - 2.

Як видно з рисунку 25, при відсутності відступів з’являються лінії на стику сегментів, що мають більшу яскравість ніж сусідні з ними пікселі. В даному випадку зображення було сегментовано на 4 рівні частини, в результаті чого, лінії перетинаються точно по центру зображення.

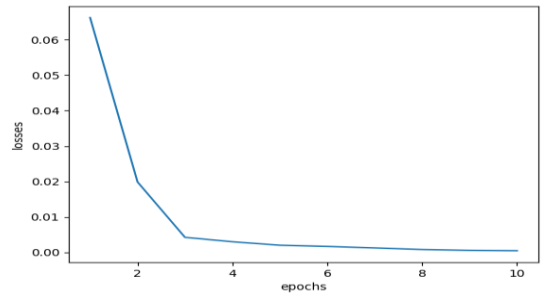


А Б

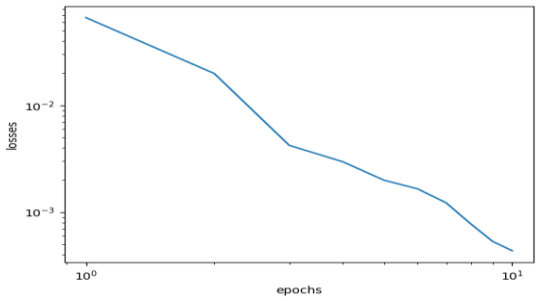
Рисунок 26 - Результати тестів: а - 3, б - 4.

Для тестів 3 і 4 (рисунок 26) було вибрано такі розміри фрагментів зображення, щоб отримати 9 сегментів. Як видно, після обробки НМ, без відступів, проявилися такі ж артефакти у вигляді ліній. У випадках, коли застосовувалися відступи, на зображеннях відсутні такі артефакти (лінії).

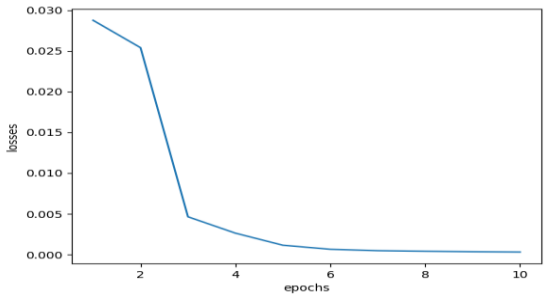
Процеси тренування представлені графіками в логарифмічному і звичайному масштабах на рисунку 27.



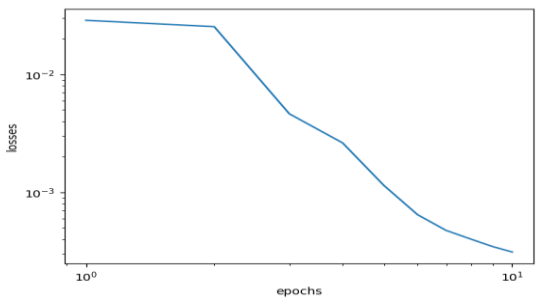
А



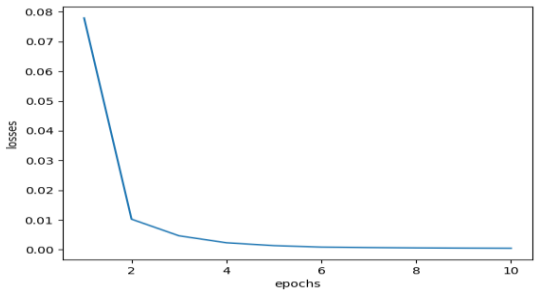
б



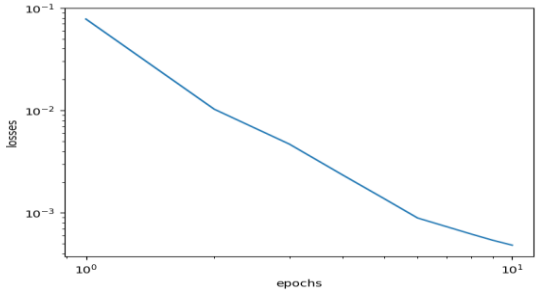
В



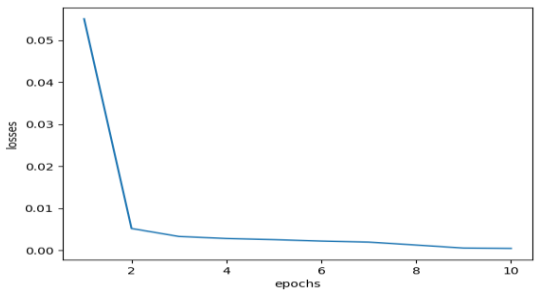
г



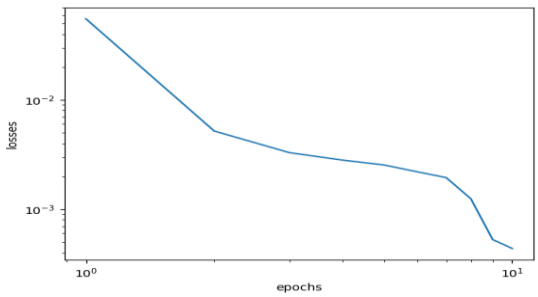
Д



е



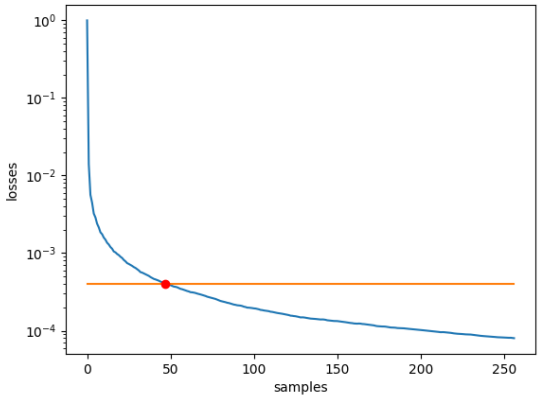
Є



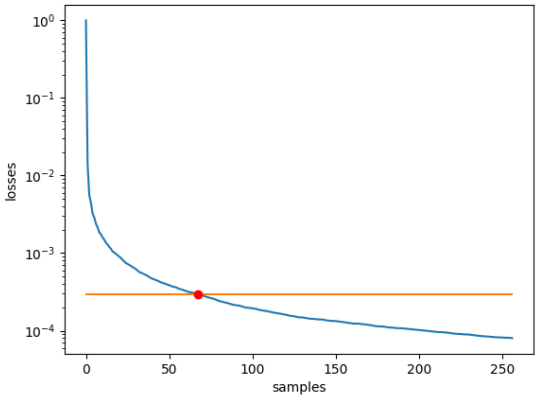
ж

Рисунок 27 - Процеси тренувань тестів 1-4 (а, б – 1; в, г – 2; д, е- 3; є, ж - 4).

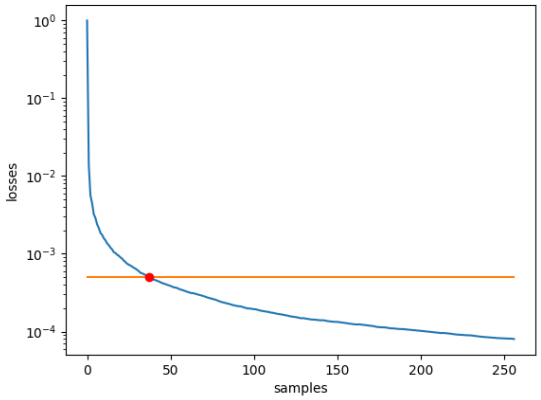
Щоб оцінити рівень шуму були побудовані графіки, що показані на рисунку 28.



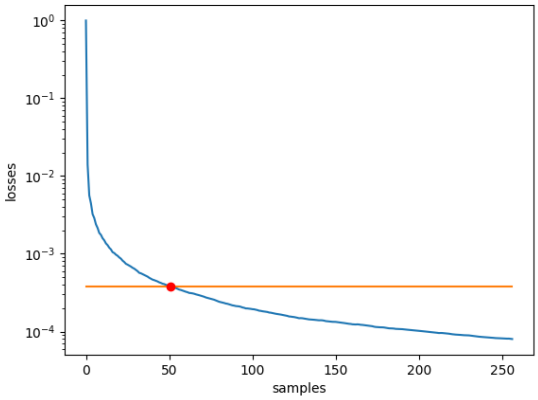
А



б



В



г

Рисунок 28 - Залежності між середньоквадратичним відхиленням і кількістю вибірок (а - 1, б - 2, в - 3, г - 4).

Як видно з графіків на рисунку 28, середнє значення відхилень від очікуваних зображень, на тестовому наборі покращилося відносно вхідних зображень (32 вибірки). Рисунки 25 і 26 демонструють, що шум було усунуто, проте з’явилися інші артефакти, які свідчать про недостатню якість тренування НМ, тобто її треба дотренувати, або натренувати з “нуля” з іншими параметрами.

4.4 Висновки за розділ

Для перевірки створеного програмного засобу застосовано критерій середньоквадратичного відхилення тренувальних (або тестових) зображень від очікуваних. Було створено 6 наборів зображень шляхом рендерингу:

1) 2048 екземплярів з 1 вибіркою;
2) 2048 екземплярів з 32 вибірками;
3) 2048 екземплярів з базовими кольорами об’єктів сцени;
4) 2048 екземплярів представлення нормалей від об’єктів сцени;
5) 2048 екземплярів з 2048 вибірками (очікувані зображення);
6) 256 екземплярів з вибірками від 1 до 256.

Порядок виконання перевірки складався з 3 етапів:

1) підготовка даних;
2) тренування НМ;
3) тестування НМ на тестовому наборі даних.

.Застосовані DAE, що були натреновані на вищезазначених наборах, підтвердили здатність знешумлювати цифрові зображення, більш подібні до очікуваних, а також перетворювати інші типи представлення зображень, при цьому, середньоквадратичне відхилення, відносно очікуваних зображень, більше ніж при знешумлені зображень Це означає, що застосована мережа найбільше підходить для знешумлення.

ВИСНОВКИ

Таким чином, задача побудови структури і підбору параметрів НМ, здатної адаптивно навчатися формуванню фільтрів для обробки зображень, вирішена.

Основою моделі зображень є модель цифрового (дискретизованого) зображення. В свою чергу, дане цифрове зображення є двовимірною проєкцією тривимірної сцени, яка згенерована алгоритмічним способом в процесі рендерингу.

Зображення отримані в процесі рендерингу, за умови не достатньої кількості часу обробки, характеризуються шумоподібною структурою, що викликано особливостями процесу рендерингу. Статистичні характеристики такого шуму за площею зображення, можуть відповідати різним розподілам (Гаусса, Релея, Ерланга, експотенційного, рівномірного, імпульсного), що визначає необхідність застосування до таких зображень процедури адаптивної фільтрації.

Таким чином, в цьому проєкті використано спеціальну ЗНМ, для знешумлення (DAE) цифрових зображень (RGB), з такими параметрами: 7 шарів згортки, однаковими об’ємами для внутрішніх шарів НМ, в якості операції проріджения використано фільтр максимуму (див. 2.2), ядра згортки в даному проєкті мають розміри 2x2 та 3x3.

Програмний засіб складається з 6 модулів. Модулі графічного інтерфейсу пов’язані між собою, та використовуються для задання параметрів підготовки даних, вказування джерел даних, тренування і тестування НМ. В разі виникнення виняткових ситуацій, програма продовжує виконуватися та дозволяє створювати (дотреновувати) декілька моделей впродовж виконання.

Для перевірки створеного програмного засобу застосовано критерій середньоквадратичного відхилення тренувальних (або тестових) зображень від очікуваних. Було створено 6 наборів зображень шляхом рендерингу:

1) 2048 екземплярів з 1 вибіркою;
2) 2048 екземплярів з 32 вибірками;
3) 2048 екземплярів з базовими кольорами об’єктів сцени;
4) 2048 екземплярів представлення нормалей від об’єктів сцени;
5) 2048 екземплярів з 2048 вибірками (очікувані зображення);
6) 256 екземплярів з вибірками від 1 до 256.

Порядок виконання перевірки складався з 3 етапів:

1) підготовка даних;
2) тренування НМ;
3) тестування НМ на тестовому наборі даних.

.Застосовані DAE, що були натреновані на вищезазначених наборах, підтвердили здатність знешумлювати цифрові зображення, більш подібні до очікуваних, а також перетворювати інші типи представлення зображень (приведення зображень до одного типу), при цьому, середньоквадратичне відхилення, відносно очікуваних зображень, більше ніж при знешумлені зображень. Це означає, що застосована мережа найбільше підходить для знешумлення.

У подальшому розвитку даного програмного засобу, основні зусилля необхідно зосередити на способі оцінки результатів знешумлення, та розбитті цього засобу на окрему програму для тренування, та окрему програму для тестування, яка буде завантажувати вагу коефіцієнтів натренованої НМ з метою знешумлення зображень користувача.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Дробик О.В., Кідалов В.В., Коваль В.В., Костік Б.Я., Лазебний В.С., Розорінов Г.М.. Цифрова обробка аудіо- та відеоінформації у мультимедійних системах: Навчальний посібник [Текст]/ Сукач Г.О. – К.: Наукова думка, 2008. – 144 с.
2. Гонсалес Р., Вудс Р. Цифровая обработка изображений. - М.:Техносфера, 2005. - 1072 с.
3. Калютов А.В. Введение в фотореалистическую графику. - СПБ.: Политехника, 2015 — 118 с.
4. Форсайт Д.А., Понс Ж. Компьютерное зрение. Современный подход: Пер. с англ. М. : Издательский дом “Вильямс”, 2004. - 928 с.
5. Литвиненко О.Н. Основы радиооптики “Техника”, 1974. – 208 с.
6. Справочник по микроэлектронной импульсной технике/ Яклвлев В.Н., Воскресенский В.В., Мирошниченко С.И. и др. – К.:Техніка, 1983. – 359 с.
7. Справочник по устройствам цифровой обработки информации/ Н.А. Виноградов, В.Н. Яковлев, В.В. Воскресенский и др. – К.:Тэхника, 1988. – 415 с.
8. Суэмацу Я., Катаока С., Кисино К., Кокобун Я., Судзуки Т., Исии О., Енедзава С. Основы оптоэлектроники: Пер. с яп. – М.: Мир, 1988. – 288 с.
9. Колесник О.Ю. Застосування нейронної мережі для стилізованої обробки зображень. Електронна та акустична інженерія. 2019. Т. 2, №4. С. 17-20.
10. PyTorch framework documatation. TORCH.NN URL: http://pytorch.org/docs/stable/nn.html (дата звернення: 11.05.2020).
11. Синєглазов В., Чумаченко О. Глибокі нейронні мережі для вирішення завдань розпізнавання і класифікації зображення

[Електронний ресурс] Синєглазов В., Чумаченко О. – 2017. – URL: http://itcm.comp-sc.if.ua/2017/Sineglazof.pdf (дата звернення: 7.05.2020).

12. Emand M. Grais, Mark D. Plumbey. Signal channel audio source separation using convolutional denoising autoencoder [Електронний ресурс] – 2017. – URL: http://core.ac.uk/download/pdf/84589145.pdf (дата звернення: 7.05.2020).
13. Kopczyk D., Denoising Autoencoder: Part I – Introduction to Autoencoders, URL: http://dkopczyk.quantee.co.uk/dae-part1 (дата звернення: 7.05.2020).
14. Xingchen L., Qicai Z., Jiong Z., Hehong S., Xiaolei X. Fault Diagnosis of Rotating Machinery under Noisy Environment Conditions Based on a 1-D Convolutional Autoencoder and 1-D Convolutional Neural Network [Електронний ресурс]. – 2019. – URL: http://mdpi.com/1424-8220/19/4/972/html (дата звернення: 7.05.2020).
15. Numpy package. About URL: https://numpy.org/about/ (дата звернення: 11.05.2020).
16. OpenCV library. About URL: https://opencv.org/about/ (дата звернення: 11.05.2020).
17. PyThorch framework. Homepage URL: https:// pytorch.org/ (дата звернення: 11.05.2020).
18. MatPlotLib library. Homepage URL: https://matplotlib.org/index.html (дата звернення: 11.05.2020).
19. Pyforms framework. Docs URL: https://pyforms.readthedocs.io/en/v4/ (дата звернення: 11.05.2020).